(19) Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 102 35 610 A1** 2004.02.19

3.

(12) # Offenlegungsschrift

(21) Aktenzeichen: **102 35 610.6**
(22) Anmeldetag: **02.08.2002**
(43) Offenlegungstag: **19.02.2004**

(51) Int Cl.[7]: **G05B 23/02**
G05B 19/04

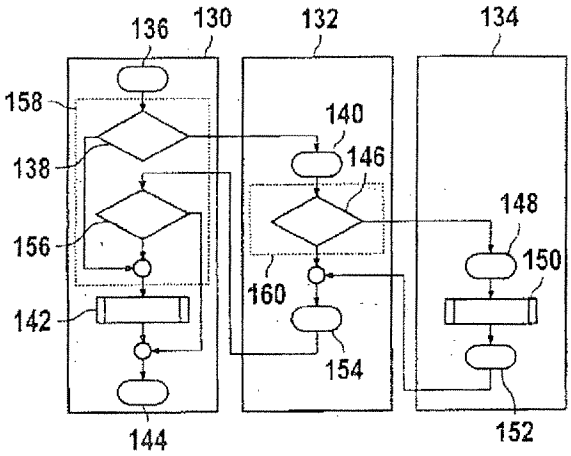
(71) Anmelder:
Robert Bosch GmbH, 70469 Stuttgart, DE

(72) Erfinder:
Watzl, Martin, 71706 Markgröningen, DE;
Nicolaou, Michael, 71665 Vaihingen, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Verfahren zum Überprüfen einer Steuergerätefunktion**

(57) Zusammenfassung: Es wird ein Verfahren zum Überprüfen einer in Software implementierten Funktion für ein Steuergerät, ein solches Steuergerät, eine Speichereinrichtung sowie ein Computerprogramm und ein Computerprogrammprodukt vorgestellt. Bei dem erfindungsgemäßen Verfahren wird die Funktion als Bypass-Programm in das Steuergeräteprogramm integriert und durch einen Aufruf zur Ausführung gebracht.



Beschreibung

[0001] Die Erfindung betrifft ein Verfahren zum Überprüfen einer in Software implementierten Funktion für ein Steuergerät und ein Steuergerät zur Durchführung des Verfahrens sowie ein Computerprogramm und ein Computerprogrammprodukt.

Stand der Technik

[0002] Zur Steuerung unterschiedlicher Funktionsbereiche in Kraftfahrzeugen werden heutzutage Steuergeräte eingesetzt. Auf dem Gebiet der Motormanagement-Systeme (Motronic) bspw. werden durch Hardware und zugeordnete Steuergerätesoftware die verschiedenen Betriebsarten eines Motors, bspw. eines Ottomotors, geregelt bzw. gesteuert. Die Einheit Steuergerät und Steuergerätesoftware übernimmt dabei in vielen Bereichen des Fahrzeugs die Ansteuerung von Aggregaten in Abhängigkeit von Eingangsgrößen.
[0003] Jede dieser Aufgaben, die von der Steuergerätesoftware bearbeitet werden muß, wird in einer Steuergerätefunktion beschrieben. Hierbei bilden eine Vielzahl von Steuergerätefunktionen sowie das Betriebssystem den sogenannten Programmstand für das Motorsteuerungssystem. Zusätzlich sind Steuergerätedaten vorgesehen, die in Form von Konstanten, applizierbaren Festwerten, Kennlinien und/oder Kennfeldern in dem betreffenden Steuergerät abgelegt sind und den Datenstand bilden. Der Programmstand und die dazugehörigen Daten bilden den Softwarestand für das Motorsteuerungssystem.
[0004] In dem Softwarestand ist durch Spezifikation der Steuergerätefunktionen und der Daten das Verhalten des Steuergeräts bei verschiedenen Betriebsarten des Motors fest definiert bzw. vorgegeben. Dabei ist möglich, mit einem geeigneten Applikationssteuergerät und dem dazugehörigen Applikationssystem durch Datenänderung in eingeschränkter Form die Funktionsweise von Steuergerätefunktionen einzustellen.
[0005] Gewöhnlich wird, wenn eine neue Steuergerätefunktionalität erzielt werden soll, an dieser Stelle ein Steuergeräte-Bypass eingesetzt, der eine Überprüfung ermöglicht, ohne daß ein neuer Programmstand generiert werden muß.
[0006] Gegenwärtig wird bspw. der sogenannte externe Steuergeräte-Bypass verwendet, der in der Funktionsentwicklung zum Einsatz kommt.
[0007] Die Entwicklungsumgebung setzt sich dabei aus einem Applikationssteuergerät mit Emulationstastkopf, einem Applikationssystem, einer externen Hardware und einem PC mit Software zur Codegenerierung zusammen. Mit der PC-Software wird die Bypass-Funktion modelliert und anschließend in Code implementiert. Der dabei erzeugte Bypass-Code wird auf eine externe Hardware geladen und dort zur Ausführung gebracht, wobei dieser über ein Softwareschalterkonzept aktiviert bzw. deaktiviert werden kann.
[0008] Auf diese Weise wird eine neue Funktionalität herbeigeführt, während gleichzeitig der vorhandene Funktionscode umgangen wird. Die dabei berechneten Bypass-Werte werden über eine Schnittstelle unter Anwendung eines Übertragungsprotokolls in die Steuergerätesoftware zurückgeschrieben.
[0009] Bei dem Verfahren mit externem Steuergeräte-Bypass erfolgt die Berechnung der Bypass-Werte außerhalb des Steuergerätesystems, wobei die Berechnung typischerweise in Float-Arithmetik durchgeführt wird. Es können auch immer nur einzelne Steuergerätegrößen mit dem Bypass bearbeitet werden. Außerdem ist das Zeitverhalten, d.h. das Lesen und Zurückschreiben von Steuergerätegrößen, fest an ein Übertragungsprotokoll gebunden. Weiterhin ist zu beachten, daß ein Freischnitt der Steuergerätefunktion und ebenso eine parallele Verwaltung der Dokumentation und Softwareerstellung der Steuergerätefunktionen mit und ohne Bypass-Freischnitt erforderlich sind.
[0010] Nachteilig bei dem Verfahren mit externem Bypass ist, daß es bei der Nachbildung von komplexen Regelkreisen zu Rechenrasterverlusten kommen kann, die das Ergebnis der , Bypass-Instrumentierung verfälschen und damit sogar unbrauchbar machen können. Außerdem wird in dem Steuergerät eine andere Arithmetik (Integer) als in der externen Hardware (Float) verwendet. Weitere Nachteile sind die bedingte Rechenzeitbeschränkung, der erhöhte Zeitaufwand für die Vorbereitung eines Bypass-Freischnitts und der hohe Verwaltungsaufwand von Steuergerätefunktionen mit und ohne Bypass-Freischnitt. Ein weiterer Nachteil ist, daß immer nur einzelne Steuergerätegrößen mit dem Bypass bearbeitet werden können.

Vorteile der Erfindung

[0011] Demgegenüber sieht das erfindungsgemäße Verfahren zum Überprüfen einer in Software implementierten Funktion für ein Steuergerät vor, daß die neue Funktion als Bypass-Programm in das Steuergeräteprogramm integriert und durch einen Aufruf zur Ausführung gebracht wird.
[0012] Mit dem erfindungsgemäßen Verfahren können die vorstehend genannten Nachteile ausgeräumt werden und die Kosten und die Entwicklungszeit bei der Softwareentwicklung verringert werden.
[0013] Bei dem erfindungsgemäßen Verfahren mit internem Steuergeräte-Bypass handelt es sich um ein Verfahren, bei dem eine neu erstellte Softwarefunktion in ein bestehendes Steuergeräteprogramm integriert und

über einen Mechanismus zur Ausführung gebracht werden kann, ohne daß eine Änderung der vorhandenen Softwarestruktur erforderlich ist. Es besteht somit die Möglichkeit, die Bypass-Funktionen in dem Zielsystem bereits während der Entwicklungsphase zu testen und auf deren softwaretechnische Tauglichkeit in Bezug auf Laufzeit, physikalisches Verhalten usw. zu überprüfen. Selbstverständlich ist es möglich, mehrere Bypass-Programme zu integrieren, von denen jedes einzelne gesondert oder in Kombination mit anderen Bypass-Programmen aufgerufen werden kann.

[0014] In Ausgestaltung der Erfindung wird das Bypass-Programm durch einen Kopiervorgang in das Steuergeräteprogramm integriert.

[0015] Bei diesem Vorgang werden die Inhalte von Speichersegmenten aus der Bypass-Software in die Steuergeräte bzw. Projekt-Software eingefügt.

[0016] Vorzugsweise ist eine Bypass-Dienstroutine vorgesehen, die über eine in einer Bypass-Pointertabelle enthaltene Adresse das Bypass-Programm aufruft.

[0017] Der Inhalt der Pointer- bzw. Zeigertabelle wird dabei durch die Bypass-Entwicklungsumgebung bestimmt. In der Tabelle wird die Zuordnung der Adressen von Steuergerätefunktionen bzw. Steuergerätegrößen zu den aufzurufenden Bypass-Funktionen definiert.

[0018] Zweckmäßigerweise wird die Bypass-Dienstroutine durch einen Softwareschalter über ein Applikationssystem aktiviert.

[0019] Eine Möglichkeit sieht vor, daß mit einem sogenannten Funktionsbypass eine Funktion des Steuergerätes manipuliert wird. Mit einem Einzelgrößenbypass kann eine Steuergerätegröße manipuliert werden.

[0020] Üblicherweise erfolgt über Referenzen eine Datenübertragung zwischen dem Steuergeräteprogramm und dem Bypass-Programm.

[0021] Das erfindungsgemäße Verfahren ermöglicht ein Rapid Prototyping im Zielsystem, da der Funktionsentwickler seine Steuergerätefunktionen unmittelbar im Zielsystem entwickeln und ohne Wirkung auf die Softwareentwicklung testen kann. Der Bypass-Code läuft dabei in Echtzeit im Steuergerät.

[0022] Das Problem von Rechenrasterverlusten tritt beim internen Steuergeräte-Bypass nicht auf, da sich die Bypass-Instrumentierung fest an die zeitlichen Vorgaben bzw. das Scheduling der Steuergerätefunktion orientiert. Im Gegensatz zum externen Steuergeräte-Bypass können Steuergerätefunktionen mit dem Funktionsbypass sowie einzelne Steuergerätegrößen mit dem Einzelgrößenbypass getestet werden. Der Verzicht auf eine teuere externe Hardware gestattet eine Reduzierung der Kosten. Des weiteren ermöglicht das erfindungsgemäße Verfahren eine Reduzierung des Verwaltungsaufwands von Steuergerätefunktionen mit und ohne Bypass-Freischnitt.

[0023] Die Entwicklungsumgebung beim erfindungsgemäßen Verfahren setzt sich aus einem Applikationssteuergerät mit beispielsweise einer Emulationseinheit, einem Applikationssystem und einem PC mit Software zur Codegenerierung zusammen.

[0024] Die erfindungsgemäße Speichereinrichtung für ein Steuergerät, in der ein Steuergeräteprogramm abgelegt ist, umfaßt eine Anzahl von Speichersegementen. Dabei ist mindestens eines der Speichersegmente für ein Bypass-Programm vorgesehen.

[0025] Vorzugsweise weist das für das Bypass-Programm vorgesehene Speichersegment einen ersten Bereich für Bypass-Daten, einen zweiten Bereich für einen Bypass-Code und einen dritten Bereich für eine Bypass-Pointertabelle auf.

[0026] Das erfindungsgemäße Steuergerät weist eine Recheneinheit und eine Speichereinrichtung auf, in der das Steuergeräteprogramm abgelegt ist. Die Speichereinrichtung umfaßt eine Anzahl von Speichersegmenten, von denen mindestens eines für ein Bypass-Programm vorgesehen ist.

[0027] Das erfindungsgemäße Computerprogramm umfaßt Programmcodemittel zum Ausführen der Schritte des vorstehend beschriebenen Verfahrens und wird auf einem Computer oder einer entsprechenden Recheneinheit durchgeführt.

[0028] Das erfindungsgemäße Computerprogrammprodukt ist auf einem computerlesbaren Datenträger gespeichert. Als geeignete Datenträger kommen EEPROMS und Flashmemories, aber auch CD-ROMS, Disketten sowie Festplattenlaufwerke zum Einsatz.

[0029] Weitere Vorteile und Ausgestaltungen der Erfindung ergeben sich aus der Beschreibung und der beiliegenden Zeichnung.

[0030] Es versteht sich, daß die vorstehend genannten und die nachstehend noch zu erläuternden Merkmale nicht nur in der jeweils angegebenen Kombination, sondern auch in anderen Kombinationen oder in Alleinstellung verwendbar sind, ohne den Rahmen der vorliegenden Erfindung zu verlassen.

Zeichnung

[0031] Die Erfindung ist anhand von Ausführungsbeispielen in der Zeichnung dargestellt und wird im folgenden unter Bezugnahme auf die Zeichnung ausführlich beschrieben.

[0032] **Fig.** 1 zeigt eine bevorzugte Ausführungsform des erfindungsgemäßen Steuergeräts in schematischer

Darstellung.

[0033] **Fig.** 2 zeigt einen Softwareentwicklungsprozeß für einen internen Steuergeräte-Bypass.

[0034] **Fig.** 3 zeigt den Aufbau einer bevorzugten Ausführungsform der erfindungsgemäßen Speichereinrichtung.

[0035] **Fig.** 4 zeigt in einer Prinzipdarstellung ein Zusammenführen von Bypass- und Steuergeräte-Software.

[0036] **Fig.** 5 zeigt einen Aufbau einer Bypass-Pointertabelle.

[0037] **Fig.** 6 zeigt einen Ablauf einer Suchfunktion in der Bypass-Dienstroutine.

[0038] **Fig.** 7 zeigt eine bevorzugte Ausführungsform des erfindungsgemäßen Verfahrens.

[0039] **Fig.** 8 zeigt eine weitere bevorzugte Ausführungsform des erfindungsgemäßen Verfahrens.

[0040] **Fig.** 9 zeigt einen Bypass-Freischnitt für den Funktionsbypass.

[0041] **Fig.** 10 zeigt einen Bypass-Freischnitt für den Einzelgrößenbypass.

[0042] In **Fig.** 1 ist eine bevorzugte Ausführungsform des erfindungsgemäßen Steuergeräts, insgesamt mit der Bezugsziffer **10** bezeichnet, dargestellt.

[0043] Das Steuergerät **10** weist eine elektronische Recheneinheit **12**, nämlich eine CPU **12**, und eine Speichereinrichtung **14** auf, die über eine Datenleitung **16** miteinander verbunden sind. Die Speichereinrichtung **14** umfaßt eine Anzahl von Speichersegmenten **18**. In der Speichereinrichtung **14** ist ein Steuerprogramm abgelegt, wobei mindestens eines der Speichersegmente **18** zur Ablage eines Bypass-Programms und somit zur Integration desselben in die Projekt-Software vorgesehen ist.

[0044] In **Fig.** 2 ist die Entwicklung der Bypass-Software parallel zur Entwicklung der Projekt-Software verdeutlicht.

[0045] In einem ersten Feld **20** ist der Entwicklungsprozeß für die Projekt-Software und in einem zweiten F **22** die Erstellung der Bypass-Software wiedergegeben.

[0046] Aus dem Entwicklungsprozeß der Projekt-Software ergeben sich in einem Schritt 24 Schnittstellen-Dateien, in einem Schritt 26 eine Linkerdatei, in einem Schritt 28 Projekt-Applikationsdaten und in einem Schritt 30 die Projekt-Software.

[0047] Bei der Erstellung der Bypass-Software wird in einem Schritt 32 ein Simulationsmodell der Bypass-Funktion erstellt und in einem Schritt 34 mit einem Codegenerator codiert. Die in Schritt 24 ermittelten Schnittstellen-Dateien werden in einem Schritt 36 als Referenzen auf Projektgrößen übergeben.

[0048] Nachfolgend wird in einem Schritt 38 der Bypass-Sourcecode erstellt und in einem Schritt 40 compiliert. Die in Schritt 26 erstellte Linkerdatei wird in einem Schritt 42 als Bypass-Linkerdatei übergeben. Die Daten werden dann mittels eines Linkers in einem Schritt 44 verbunden. In einem Schritt 46 wird dann die Bypass-Software und in einem Schritt 48 werden die Bypass-Applikationsdaten erhalten.

[0049] Am Ende des Entwicklungsprozesses erhält der Anwender in einem Schritt 50 die Projekt-Software und in einem Schritt 52 die Projekt-Applikationsdatei inklusive Bypass-Applikationsdaten.

[0050] Die Entwicklung der Bypass-Software erfolgt somit unabhängig von der Entwicklung der Projekt-Software. Aus dem Entwicklungsprozeß der Projekt-Software werden Schnittstellen-Dateien dem Bypass-Entwicklungsprozess übergeben, die für den Zugriff auf Daten der Projekt-Software benötigt werden.

[0051] In **Fig.** 3 ist der Aufbau einer erfinddungsgemäßen Speichereinrichtung **60** verdeutlicht. Ein erster Bereich **62** ist als Bypass-RAM-Bereich für die globalen Bypass-Größen, ein zweiter Bereich **64** als Bypass-C- de-Bereich für den Bypass-Programmcode, ein dritter Bereich **66** für Pointertabellen für den Funktionsbypa , ein vierter Bereich **68** für Pointertabellen für den Einzelgrößenbypass und ein fünfter Bereich **70** als Datenbereich für die Bypass-Applikationsdaten vorgesehen.

[0052] Für das Verfahren des internen Steuergeräte-Bypasses werden Speicherbereiche für die Bypass-Instrumentierung in der Steuergeräte-Software benötigt. Diese werden durch die Bypass-Entwicklungsumgebung mit Daten gefüllt. Dabei verhindern die Speichersegmente eine Adreßverschiebung durch die Nachbearbeitung. Aus diesem Grund bleibt die vorhandene Steuergerätesoftware weitgehend in deren Originalzustand.

[0053] In der Projekt-Software gibt es somit für globale Bypass-RAM-Größen den Bypass-RAM-Bereich. Für Steuergerätedaten, wie bspw. Festwerte, Kennlinien oder Kennfelder, die in der Bypass-Funktion als Applikationsdaten definiert werden, ist der Bypass-Datenbereich vorgesehen. Der eigentliche Bypass-Programmcode liegt im Speichersegment Bypass-Code vor.

[0054] Weiterhin werden noch die Bereiche für die Bypass-Pointertabellen benötigt.

[0055] In **Fig.** 4 ist prinzipiell das Zusammenführen von Bypassund Steuergeräte- bzw. Projekt-Software wiedergegeben. Dargestellt ist ein RAM-Bereich **80**, in dem die Projekt-Software abgelegt ist. Pfeil **82** verdeutlicht, daß die in der Bypass-Funktion spezifizierten RAM-Zellen in der Projekt-Software den Bypass-RAM-Bereich belegen.

[0056] Ein weiterer Bereich **84** stellt den Bypass-Codebereich **84** dar. Der Bypass-Code wird dabei in den Bypass-Codebereich **84** kopiert. Ein Bereich **86** dient der Aufnahme der Pointertabelle für den Funktionsbypass und noch ein weiterer Bereich **88** der Aufnahme der Pointertabelle für den Einzelgrößenbypass. Schließlich befinden sich in einem Bereich **90** die Bypass-Daten. In diesem Bereich **90** sind die Applikationsdaten enthalten, die in der Bypass-Funktion definiert sind.

[0057] Die Bypass-Software wird mittels der Bypass-Entwicklungsumgebung erzeugt. Aufbau und Struktur der Speichersegmente stimmen mit denen der Projekt-Software überein. Durch einen mit Pfeilen **92** gekennzeichneten Kopiervorgang (Delta-Flashen) werden die Inhalte der Speichersegmente aus der Bypass-Software in die Projekt-Software eingefügt und damit in diese integriert.

[0058] In **Fig.** 5 ist ein möglicher Aufbau einer Bypass-Pointertabelle **100** dargestellt. In einer ersten Spalte **102** ist ein Tabellenindex enthalten. In einer zweiten Spalte **104** befinden sich die Adressen der Steuergerätefunktionen bzw. Steuergerätegrößen. Dabei bestimmt beim Einzelgrößenbypass der Datentyp der zu manipulierenden Steuergerätegröße die Datenstruktur der Tabelle. In einer dritten Spalte **106** sind die Funktionspointeradressen der aufzurufenden Bypass-Funktionen enthalten.

[0059] Die Bypass-Pointertabelle **100** wird von der Bypass-Dienstroutine verwendet. Der Inhalt der Tabelle 100 wird durch die Bypass-Entwicklungsumgebung bestimmt. In der Tabelle 100 wird die Zuordnung der Adressen von Steuergerätefunktionen bzw. Steuergerätegrößen festgelegt.

[0060] Zu beachten ist, daß es grundsätzlich zwei Arten von Pointertabellen **100** gibt, nämlich eine für den Funktionsbypass und eine weitere für den Einzelgrößenbypass. Die Tabellen **100** haben jeweils eine feste Länge und Datenstruktur.

[0061] Nachfolgend ist beispielhaft die Umsetzung einer Bypass-Pointertabelle für den Funktionsbypass aufgeführt:

```
#define    ibTskTabLen          10


const struct
  {
    void (*src)(void);
    void (*tsk)(void);
  } ibTskTab[ibTskTabLen] = {NULL, NULL};
```

Entsprechend für den Enzelgrößenbypass:

```
#define    ibSint8TabLen          10


const struct
  {
    sint8 *src;
    sint8 (*tsk)(sint8 *srcAdr);
  } ibSint8Tab[ibSint8TabLen] = (NULL, NULL);
```

[0062] In **Fig.** 6 ist ein Ablauf einer Suchfunktion in der Bypass-Dienstroutine dargestellt. In einem Schritt erfolgt der Start der Suchroutine. Anschließend wird in einem Schritt 112 ein Parameter i gleich Null gesetzt. Dann erfolgt in einem Schritt 114 die Überprüfung, ob ein Tabellenwert gleich dem Wert des Parameters ist. Ist dies der Fall, wird in einem Schritt 116 der entsprechende Bypass-Funktionscode aufgerufen.

[0063] Wird in dem Schritt 114 festgestellt, daß der Tabellenwert nicht gleich dem Wert des Parameters ist, wird in einem Schritt 118 überprüft, ob der Tabellenwert gleich Null ist. Ist dies nicht der Fall, wird der Parameter i in einem Schritt 120 um 1 erhöht. Anschließend wird in einem Schritt 122 überprüft, ob i gleich n ist. Ist dies nicht der Fall erfolgt ein Sprung zu Schritt 114.

[0064] Falls i gleich n oder wenn in Schritt 118 festgestellt wird, daß der Tabellenwert gleich Null ist, erscheint in Schritt 124 eine Fehlermeldung, da keine Bypass-Funktion spezifiziert ist. Mit Schritt 126 endet die Suchroutine.

[0065] Die Bypass-Dienstroutine, die die Schnittstelle zwischen Steuergerätefunktion und Bypass-Funktion darstellt, ist Bestandteil des Betriebssystems. Als Übergabeparameter wird dieser beim Funktionsbypass die Adresse der aufgerufenen Steuergerätefunktion und beim Einzelgrößenbypass die Adresse der zu manipulierenden Steuergerätegröße übergeben.

[0066] Die Bypass-Dienstroutine umfaßt einen Suchalgorithmus, der in der entsprechenden Pointertabelle nach dem Wert des Übergabeparameters sucht. Wird dieser gefunden, erfolgt der Aufruf der operativen Bypass-Funktion über den Funktionspointer. Dabei gibt es jeweils Dienstroutinen für den Funktionsbypass und für den Einzelgrößenbypass. Diese unterscheiden sich in der Bearbeitung der Pointertabellen, in der Datenstruktur und in der Behandlung der Rückgabewerte.

[0067] Nachfolgend ist ein Beispiel einer Bypass-Dienstroutine für den Funktionsbypass gegeben:

```
int callIbTsk(void (srcAdr)(void))
{
int i;

for (I=0; ibTskTab[i].src != NULL && i < ibTskTabLen; i++)
  {
   if (if (ibTskTab[i].src == srcAdr)
     {
      if (ibTskTab[i].tsk !=NULL)
       {
        if(ibTskTab[i].src == ibTskTab[i].tsk)
         {
          return 1;
         }
        else
         {
           (*ibTskTab[i].tsk)();
           return 0;
         }
       }

     }
   }
return 2;
}
```

[0068] Nachfolgend ist ein Beispiel für eine Bypass-Dienstroutine für den Einzelgrößenbypass aufgeführt:

```
sint8 getIbSint8(sint8 *srcAdr)
{
int i;

for (i=0; ibSint8Tab(i).src ! = NULL && i < ibSint8TabLen;
i++)

  {
     if (ibSint8Tab(i).src = = src Adr)
       {
         if (ibSint8Tab(i).tsk ! = NULL)
          {
            return (*ibSint8Tab(i).tsk) (srcAdr);
          }
         else
          {
            return *srcAdr;
          }
       }
     }
  return *srcAdr;
}
```

[0069] Bei der Datenübertragung von Projekt- zu Bypass-Software ist zu beachten, daß die Bypass-Funktion Dateninformationen aus der Projekt-Software benötigt. Der Zugriff auf Projektgrößen kann nur über Referenzen auf diese Größen erfolgen. Es kann somit nur über Adreßinformationen und nicht über symbolische Namen der RAM-Zellen, auf Kenngrößen oder Bibliotheksfunktionen zugegriffen werden, da das Zusammenfügen von Projekt-Software und Bypass-Software nicht über den Zinkvorgang erfolgt.

[0070] In **Fig.** 7 ist eine bevorzugte Ausführung des erfindungsgemäßen Verfahrens zur Erläuterung des Funktionsbypasses gezeigt. Die Darstellung verdeutlicht die Funktionsweise des Aufrufs einer Bypass-Funktion aus einer Steuergerätefunktion für den Funktionsbypass.

[0071] Ein erster Bereich **130** zeigt die Abläufe bei der Steuergerätefunktion, ein zweiter Bereich **132** die Abläufe in der Bypass-Dienstroutine und ein dritter Bereich **134** diejenigen in der Bypass-Funktion für den Funktionsbypass.

[0072] Mit einem Schritt 136 erfolgt der Start. Anschließend wird in einem Schritt 138 überprüft, ob der Schalter für eine Bypass-Aktivierung gesetzt ist. Ist dies der Fall, erfolgt in einem Schritt 140 der Start der Bypass-Dienstroutine. Anderenfalls wird in einem Schritt 142 der Steuergerätecode ausgeführt und in einem Schritt 144 der Vorgang beendet.

[0073] Nach Start der Bypass-Dienstroutine mit Schritt 140 wird in einem weiteren Schritt 146 überprüft, ob die Adresse der Steuergerätefunktion in der Pointertabelle vorhanden ist. Ist dies der Fall, erfolgt in einem Schritt 148 der Start für den Funktionsbypass und es wird in einem Schritt 150 der Bypass-Funktionscode ausgeführt. In einem Schritt 152 endet die Ausführung des Bypass-Funktionscodes. Anschließend endet in einem Schritt 154 ebenso die Ausführung der Bypass-Dienstroutine.

[0074] Wird in dem Schritt 146 festgestellt, daß die Adresse der Steuergerätefunktion in der Pointertabelle

nicht vorhanden ist, erfolgt mit Schritt 154 die Beendigung des Ablaufs der Bypass-Dienstroutine.

[0075] Nach Beendigung der Bypass-Dienstroutine in Schritt 154 wird in einem Schritt 156 überprüft, ob die Bypass-Ausführung plausibel ist. Ist dies der Fall, erfolgt ein Sprung zu Schritt 144. Anderenfalls erfolgt ein Sprung zu Schritt 142.

[0076] In der Figur ist mit einem ersten Feld **158** ein Bypass-Freischnitt für den Funktionsbypass und mit einem zweiten Feld **160** der Suchalgorithmus gekennzeichnet.

[0077] Wie die Darstellung verdeutlicht, befindet sich in der Steuergerätefunktion der Bypass-Freischnitt. Die Bypass-Dienstroutine wird durch einen Softwareschalter über das Applikationssystem aktiviert. Dieser wird als Parameter die Adresse der aufgerufenen Steuergerätefunktion übergeben. Dann wird mit einem Suchalgorithmus in der Bypass-Pointertabelle nach dem übergebenen Parameterwert gesucht. Für den Fall, daß keine vergleichbare Adresse gefunden werden kann oder daß die Tabelle leer ist, wird auch keine Bypass-Funktion aufgerufen. Befindet sich eine passende Adresse in der Tabelle, wird über den zugeordneten Funktionspointer die Bypass-Funktion aktiviert. Dann kommt in der Bypass-Funktion die neue Steuergeräte-Funktionalität zum Einsatz. Nach erfolgreicher Beendigung dieser Aktion, liefert die Dienstroutine ein "OK" an die Steuergerätefunktion zurück und der eigentliche Steuergerätecode wird umgangen. Im Fehlerfall jedoch wird dieser ausgeführt.

[0078] **Fig.** 8 verdeutlicht das Prinzip des Einzelgrößen-Bypasses. Ein erster Bereich **170** zeigt die Abläufe in der Steuergerätefunktion, ein zweiter Bereich **172** die Abläufe in der Bypass-Dienstroutine für den Einzelgrößenbypass und ein dritter, Bereich diejenigen in der Bypass-Funktion für den Einzelgrößenbypass.

[0079] Mit Schritt 176 beginnt der Vorgang. Anschließend wird in einem Schritt 178 überprüft, ob der Schalter für die Bypass-Aktivierung gesetzt ist. Ist dies der Fall, startet in einem Schritt 180 die Bypass-Dienstroutine. Ist dies nicht der Fall bleibt die Steuergerätegröße unverändert (Schritt 182). In einem Schritt 184 wird d der Steuergerätefunktionscode ausgeführt und in Schritt 186 die Aktion beendet.

[0080] Nach Start der Bypass-Dienstroutine in Schritt 180 wird in Schritt 188 überprüft, ob die Adresse der Steuergerätegröße in der Pointertabelle vorhanden ist. Ist die Größe in der Tabelle vorhanden, beginnt in Schritt 190 die Ausführung der Bypass-Funktion für den Einzelgrößenbypass. In Schritt 192 erfolgt dann die Ausführung des Bypass-Funktionscodes. In Schritt 194 wird als Rückgabewert der berechnete Bypass-Wert übergeben und in Schritt 196 endet die Ausführung der Bypass-Funktion. Der Rückgabewert ist gleich dem Rückgabewert der Bypass-Funktion (Schritt 198). In Schritt 200 endet dann die Ausführung der Bypass-Dienstroutine.

[0081] Wird in Schritt 188 festgestellt, daß die Adresse der Steuergerätegröße nicht in der Pointertabelle vorhanden ist, wird als Rückgabewert der Parameterwert übergeben (Schritt 202) und anschließend mit Schritt 200 die Ausführung der Bypass-Dienstroutine beendet.

[0082] Nach Beendigung der Bypass-Dienstroutine in Schritt 200 wird in Schritt 204 die Steuergerätegröße gleich dem Rückgabewert der Bypass-Dienstroutine gesetzt und mit Schritt 184 fortgefahren.

[0083] Zusätzlich sind in der Darstellung mit einem Feld **206** der Bypass-Freischnitt für den Einzelgrößenbypass und mit einem Feld **208** der Suchalgorithmus gekennzeichnet.

[0084] Wie die Figur verdeutlicht, funktioniert der Einzelgrößenbypass prinzipiell ähnlich wie der Funktionsbypass. In einer Steuergerätefunktion befindet sich ein Bypass-Freischnitt. Die Bypass-Dienstroutine wird durch einen Softwareschalter über das Applikationssystem aktiviert. Dieser wird als Parameter die Adresse der zu manipulierenden Steuergerätegröße übergeben. Dann wird durch einen Suchalgorithmus in der Bypass-Pointertabelle nach dem übergebenen Parameterwert gesucht. Wird keine vergleichbare Adresse gefunden oder ist die Tabelle leer, wird auch keine Bypass-Funktion aufgerufen. Befindet sich eine passende Adresse in der Tabelle, wird über den zugeordneten Funktionspointer die Bypass-Funktion aktiviert. In der Bypass-Funktion kommt dann die neue Steuergerätefunktionalität zum Einsatz.

[0085] Nach erfolgreichem Abschluß dieser Aktion, liefert die Bypass-Funktion einen Rückgabewert an die Dienstroutine. Der Rückgabewert wird an die Steuergerätefunktion weitergegeben. Mit diesem wird anschließend die entsprechende RAM-Zelle beschrieben. Im Fehlerfall ist der Rückgabewert der Dienstroutine der übergebene Parameterwert, weshalb sich der Wert der Steuergerätegröße nicht ändert.

[0086] Die Bypass-Funktion für den Funktionsbypass ist wie eine herkömmliche Steuergerätefunktion aufgebaut. Der Zugriff auf Projektgrößen erfolgt typischerweise über Referenzen auf die zu lesenden bzw. die zu manipulierenden Steuergerätegrößen. Darüber hinaus ist die Bypass-Funktion ausgelegt, Bibliotheksfunktionen, wie regelungstechnische Übertragungsglieder und Interpolationsroutinen, oder Applikationsdaten aus der Projekt-Software zu verwenden. Projektgrößen können nur durch deren Adressen, nicht aber durch das Label, dem Bypass zugänglich gemacht werden.

[0087] Nachfolgend ist beispielhaft eine Bypass-Funktion für den Funktionsbypass aufgeführt:

```
void iBzwstt_10ms(void)
{
zwstt = xyz
}
```

[0088] Die Bypass-Funktion für den Einzelgrößenbypass ist ähnlich aufgebaut. Einziger Unterschied ist, daß die Funktion einen typisierten Rückgabewert hat. Des weiteren wird der Rückgabewert auf die entsprechende Steuergerätegröße zurückgeschrieben.

[0089] Nachfolgend ist beispielhaft eine Bypass-Funktion für den Einzelgrößenbypass aufgeführt:

```
sint8 iBwdkba_w(sint8 * srcAdr)
{
return IBFWWDKBA_W;
}
```

[0090] **Fig.** 9 zeigt in einer Prinzipdarstellung einen Bypass-Freischnitt für den Funktionsbypass.

[0091] In Schritt 210 beginnt der Vorgang. In Schritt 212 wird überprüft, ob Bit **1** und Bit **3** des Codeworts gesetzt sind. Ist dies der Fall, erfolgt in Schritt 214 die Ausführung der Bypass-Dienstroutine. Dann wird in einem Schritt 216 überprüft, ob der Rückgabewert der Dienstroutine kleiner als 2 ist. Ist dies der Fall, endet die Aktion in einem Schritt 218.

[0092] Wird in Schritt 212 festgestellt, daß Bit **1** und Bit **3** nicht gesetzt sind, oder wird in Schritt 216 festgestellt, daß der Rückgabewert der Dienstroutine nicht kleiner als 2 ist, wird in einem Schritt 220 der Steuergerätefunktionscode ausgeführt.

[0093] Beim Bypass-Freischnitt für den Funktionsbypass befindet sich in einer Steuergerätefunktion ein Bypass-Freischnitt. Dieser Freischnitt ist ein Schaltergerüst aus Kenngrößen, die über das Applikationssystem geändert werden können. Dadurch wird die Codesequenz für den Bypass aktiviert. Als Wert liefert die Dienstroutine den Status der Bypass-Ausführung zurück, der im Freischnitt plausibilisiert wird.

[0094] Nachfolgend ist ein Beispiel einer Steuergerätefunktion mit Bypass-Freischnitt für den Funktionsbypass gegeben: void zwstt l0ms(void)

```
 {
if (GETBIT(CWIBENZWSTT, 1) && GETBIT(CWIBENZWSTT, 3))
  {
   if ((iBzwstt_10ms-status = callIbTsk(zwstt_10ms))<2)
   {
    return;
    }
 }
else
{
// ...
// ... Steuergerätefunktionscode ...
// ...
zwstt = xyz
}
```

[0095] In **Fig.** 10 ist ein Bypass-Freischnitt für den Einzelgrößenbypass gezeigt. Mit einem Schritt 230 beginnt die Aktion. In einem Schritt 232 wird überprüft, ob Bit **0** des Codeworts gesetzt ist. Ist dies der Fall wird in einem Schritt 234 die Bypass-Dienstroutine ausgeführt. Mit einem Schritt 236 wird die Steuergerätegröße gleich dem Rückgabewert der Bypass-Dienstroutine gesetzt. In einem Schritt wird dann mit der Ausführung des Steuergerätefunktionscodes fortgefahren bis zum Ende des Vorgangs in einem Schritt 240.

[0096] Wird in Schritt 232 festgestellt, daß Bit **0** nicht gesetzt ist, bleibt die Steuergerätegröße unverändert (Schritt 242) und es erfolgt ein Sprung zu Schritt 238.

[0097] Der Aufbau des Freischnitts beim Einzelgrößenbypass ist ähnlich demjenigen beim Funktionsbypass. Einziger Unterschied ist, daß die Dienstroutine den Wert aus der Bypass-Funktion auf die entsprechende Steuergerätegröße in der Steuergerätefunktion schreibt.

[0098] Nachfolgend ist noch ein Beispiel einer Steuergerätefunktion mit Bypass-Freischnitt für den Einzelgrößenbypass aufgeführt:

```
void zwwl_20ms (void)
{
//
// ... Steuergerätefunktionscode ...
.. ...
if (GETBIT(CWIBENDZWWL, 0))
  {
   dzwwl = getIbSint8(6dzwwl);
   }
else
   {
    dzwwl = xyz;
    }
    ..
    ... Steuergerätefunktionscode ...
    ...
  }
```

Patentansprüche

1. Verfahren zum Überprüfen einer in Software implementierten Funktion für ein Steuergerät (**10**), in dem ein Steuergeräteprogramm abgelegt ist, bei dem die Funktior als Bypass-Programm in das Steuergeräteprogramm integriert und durch einen Aufruf zur Ausführung gebracht wird.

2. Verfahren nach Anspruch 1, bei dem das Bypass-Programn durch einen Kopiervorgang in das Steuergeräteprogramm integriert wird.

3. Verfahren nach Anspruchs 1 oder 2, bei dem eine Bypass-Dienstroutine vorgesehen ist, die über eine in einer Bypass-Pointertabelle (**100**) enthaltenen Adresse das Bypass-Programm aufruft.

4. Verfahren nach Anspruch 3, bei dem die Bypass-Dienstroutine durch einen Softwareschalter über ein Applikationssystem aktiviert wird.

5. Verfahren nach einem der Ansprüche 1 bis 4, bei dem eine Steuergerätefunktion manipuliert wird.

6. Verfahren nach einem der Ansprüche 1 bis 4, bei dem eine Steuergerätegröße manipuliert wird.

7. Verfahren nach einem der Ansprüche 1 bis 6, bei dem über Referenzen eine Datenübertragung zwischen dem Steuergeräteprogramm und dem Bypass-Programm durchgeführt wird.

8. Speichereinrichtung für ein Steuergerät (**10**) mit einer Anzahl von Speichersegmenten (**18**), in dem ein Steuergeräteprogramm abgelegt ist, wobei mindestens eines der Speichersegmente (**18**) für ein Bypass-Programm vorgesehen ist.

9. Speichereinrichtung nach Anspruch 8, bei dem das mindestens eine für das Bypass-Programm vorgesehene Speichersegment (**18**) einen ersten Bereich für Bypass-Daten, einen zweiten Bereich für einen Bypass-Code und einen dritten Bereich für eine Bypass-Pointertabelle (**100**) aufweist.

10. Steuergerät mit einer Recheneinheit (**12**) und einer Speichereinrichtung (**14**), die eine Anzahl von Speichersegmenten (**18**) aufweist und in der ein Steuergeräteprogramm abgelegt ist, wobei mindestens eines der Speichersegmente für ein Bypass-Programm vorgesehen ist.

11. Computerprogramm mit Programmcodemitteln, um alle Schritte eines Verfahrens nach einem der Ansprüche 1 bis 7 durchzuführen, wenn das Computerprogramm auf einem Computer oder einer entsprechenden Recheneinheit (**12**), insbesondere einer elektronischen Recheneinheit (**12**) in einem Steuergerät (**10**) nach Anspruch 10, ausgeführt wird.

12. Computerprogrammprodukt mit Programmcodemitteln, die auf einem computerlesbaren Datenträger gespeichert sind, um ein Verfahren nach einem der Ansprüche 1 bis 7 durchzuführen, wenn das Computerprogramm auf einem Computer oder einer entsprechenden Recheneinheit (**12**), insbesondere einer elektronischen Recheneinheit (**12**) in einem Steuergerät (**10**) nach Anspruch 10, ausgeführt wird.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

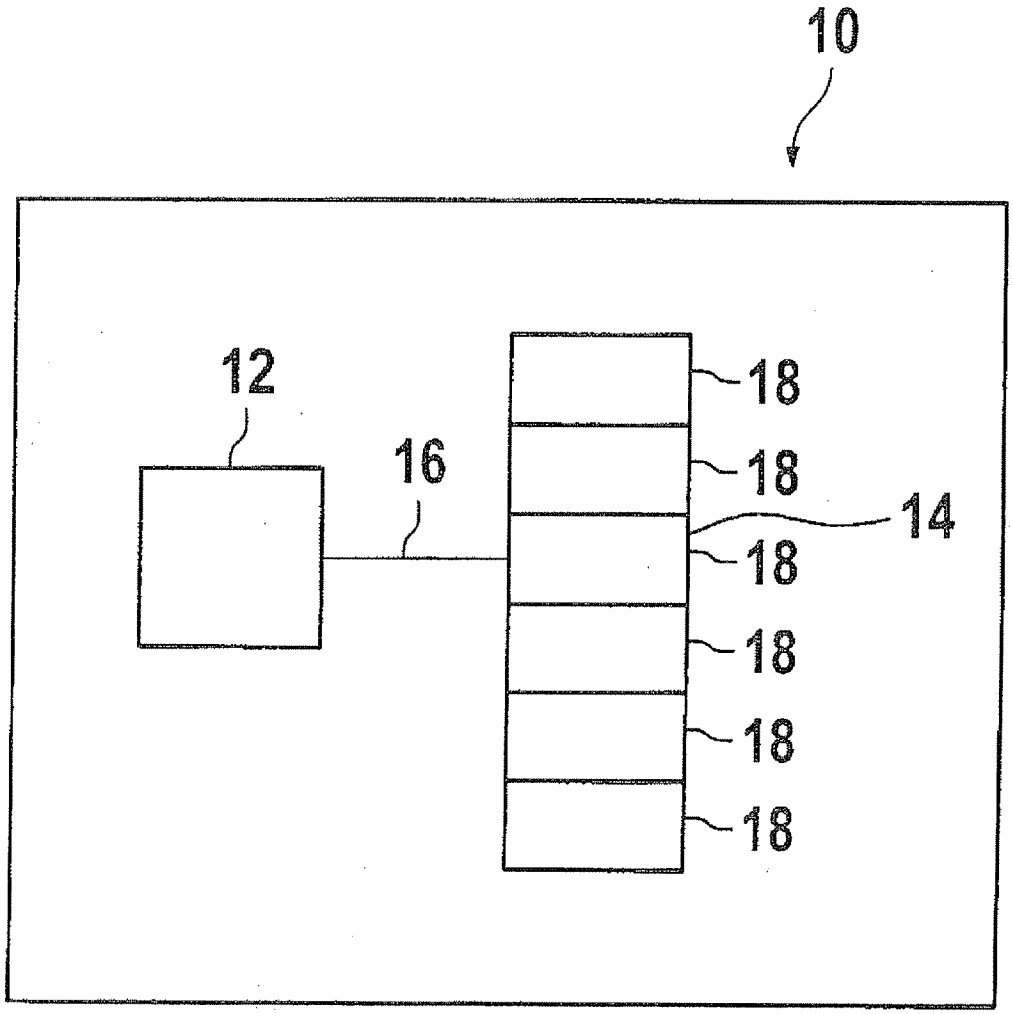


FIG. 1

FIG. 2

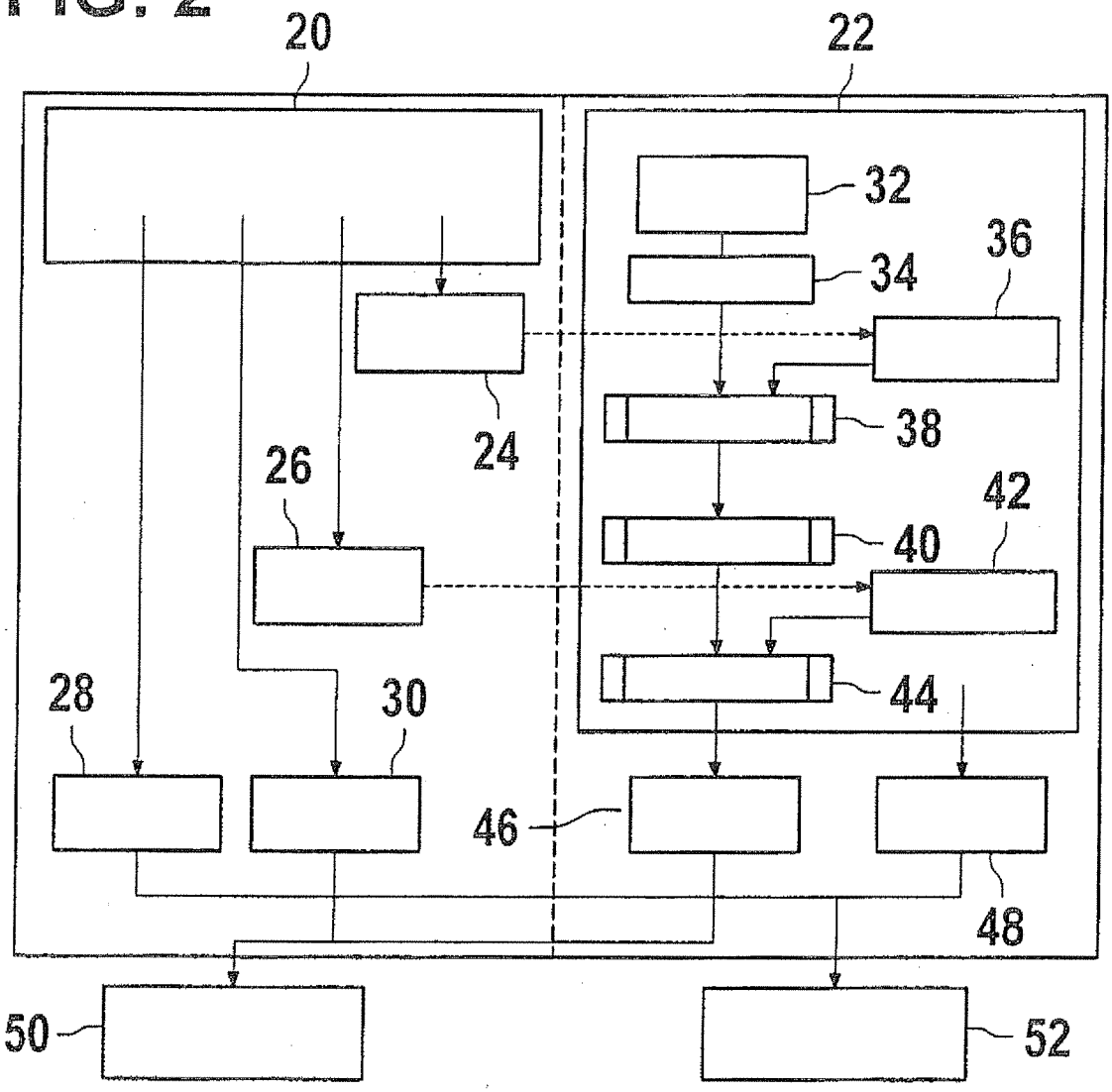

20
22
32
34
36
24
38
26
42
40
28
30
44
46
48
50
52

FIG. 3

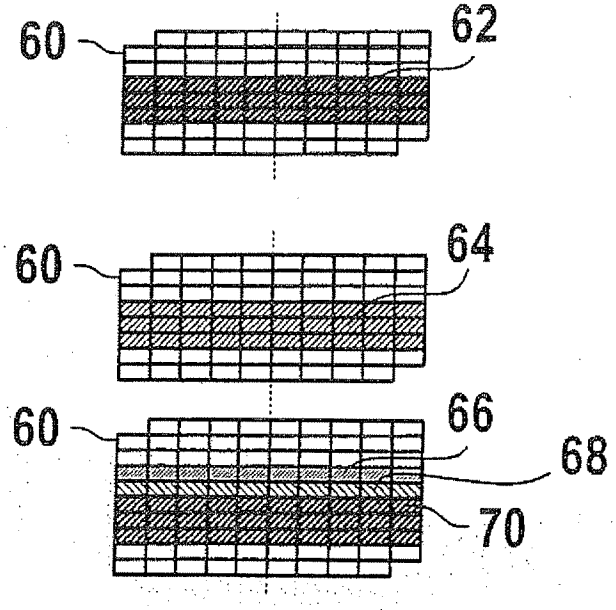

60
62
60
64
60
66
68
70

FIG. 4

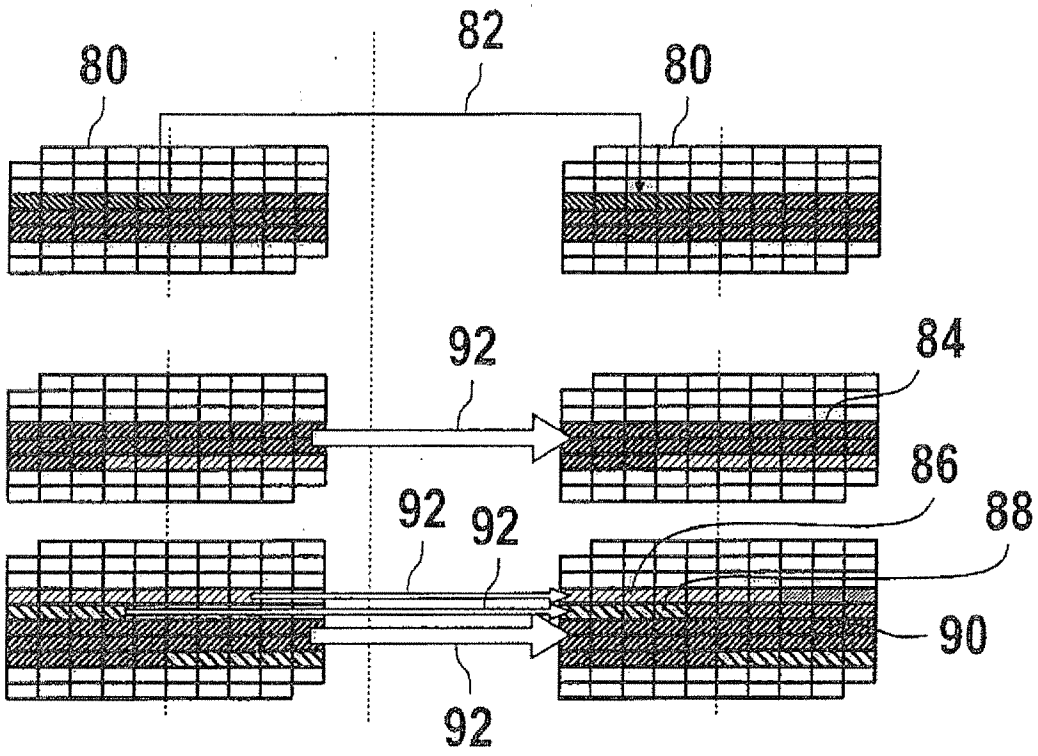

82
80
80
92
84
86
92
92
88
90
92

FIG. 5

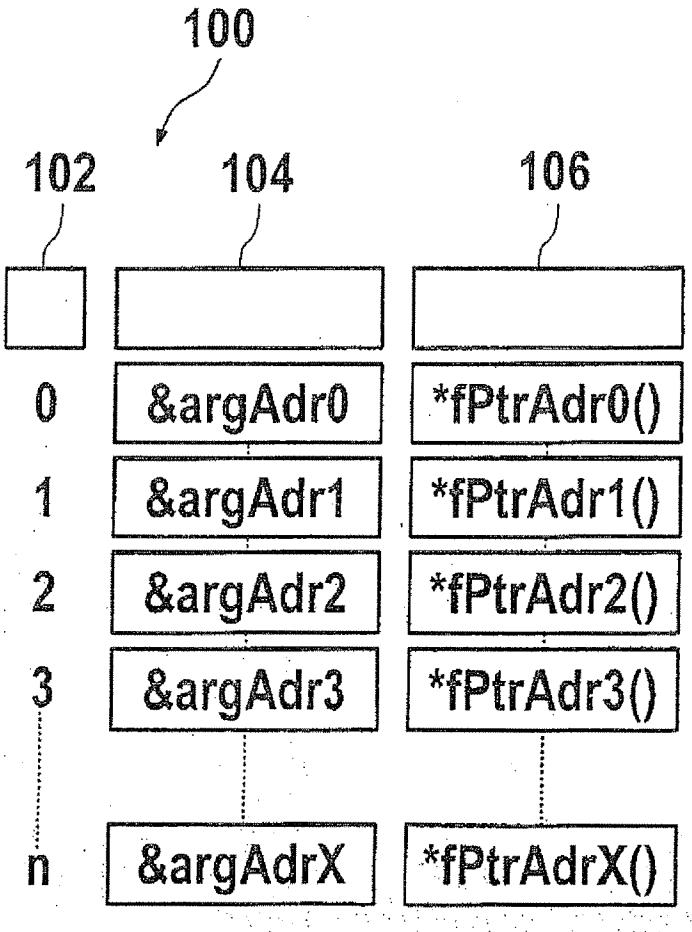

100
102
104
106
0
&argAdr0
*fPtrAdr0()
1
&argAdr1
*fPtrAdr1()
2
&argAdr2
*fPtrAdr2()
3
&argAdr3
*fPtrAdr3()
n
&argAdrX
*fPtrAdrX()

FIG. 6

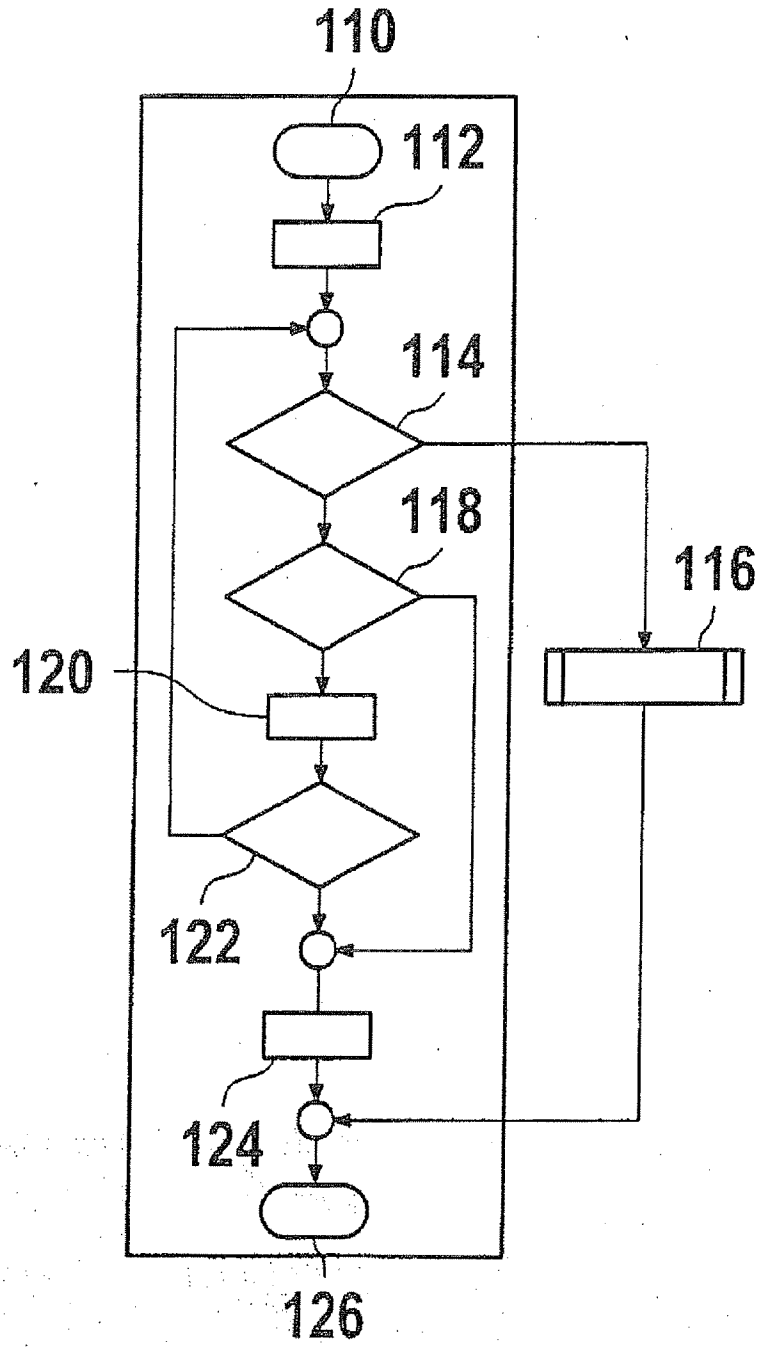


FIG. 7

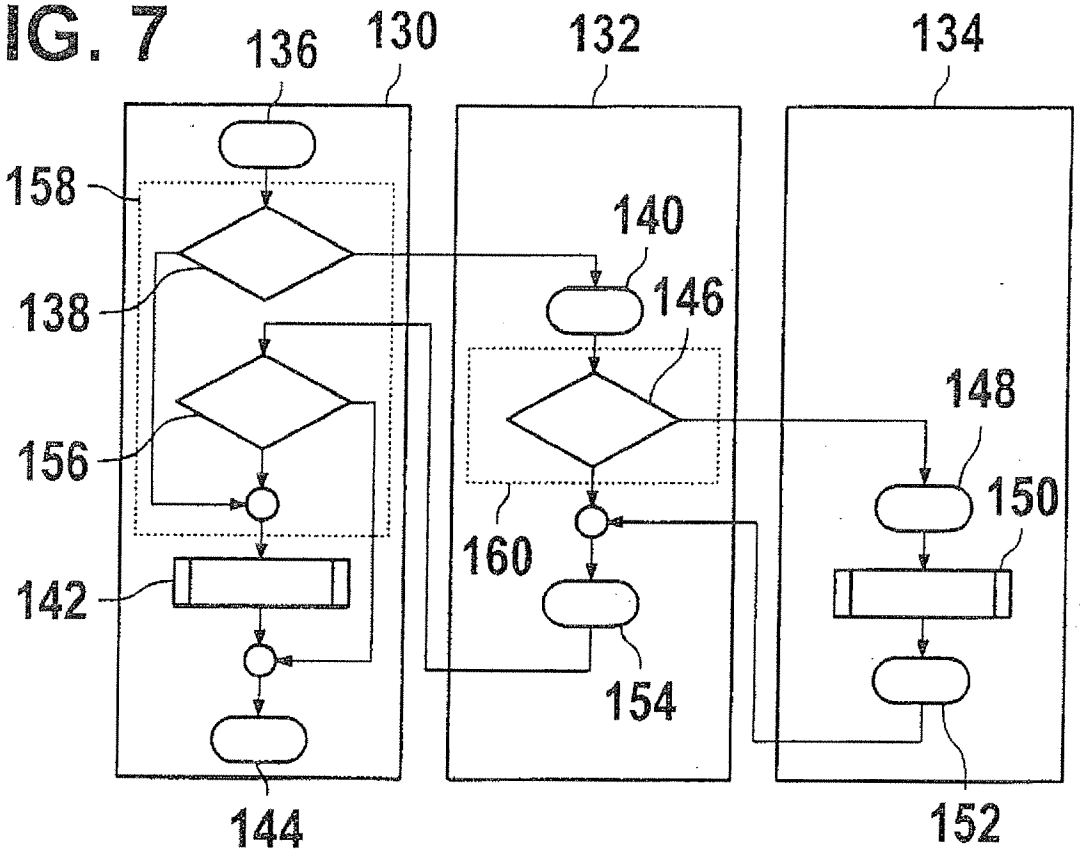

136
130
132
134
158
140
138
146
148
156
150
160
142
154
144
152

FIG. 8

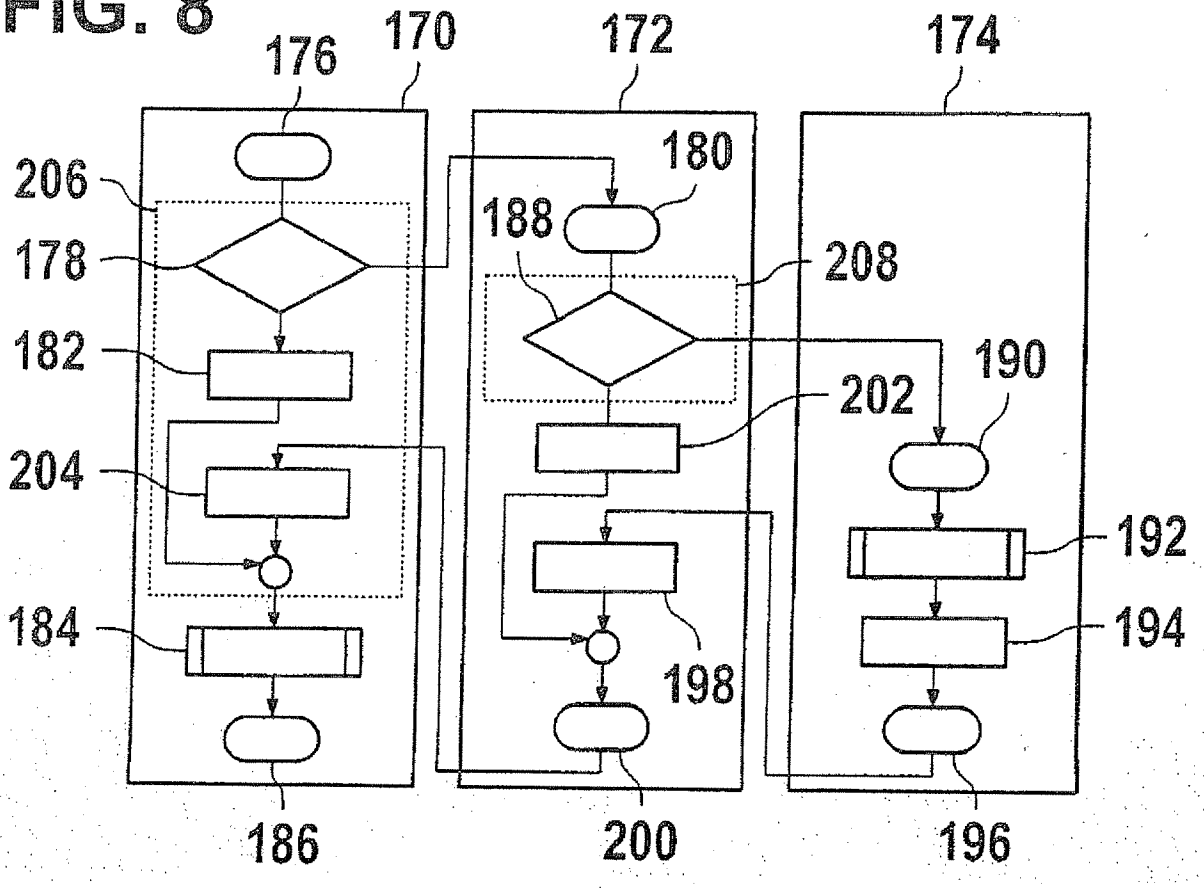

176
170
172
174
206
180
188
178
208
182
190
202
204
192
184
194
198
186
200
196

FIG. 9

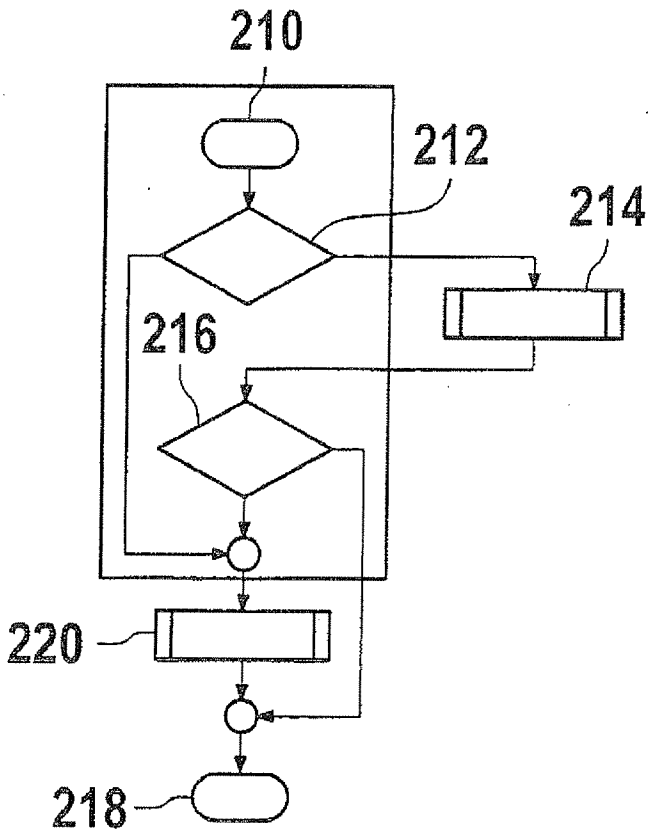
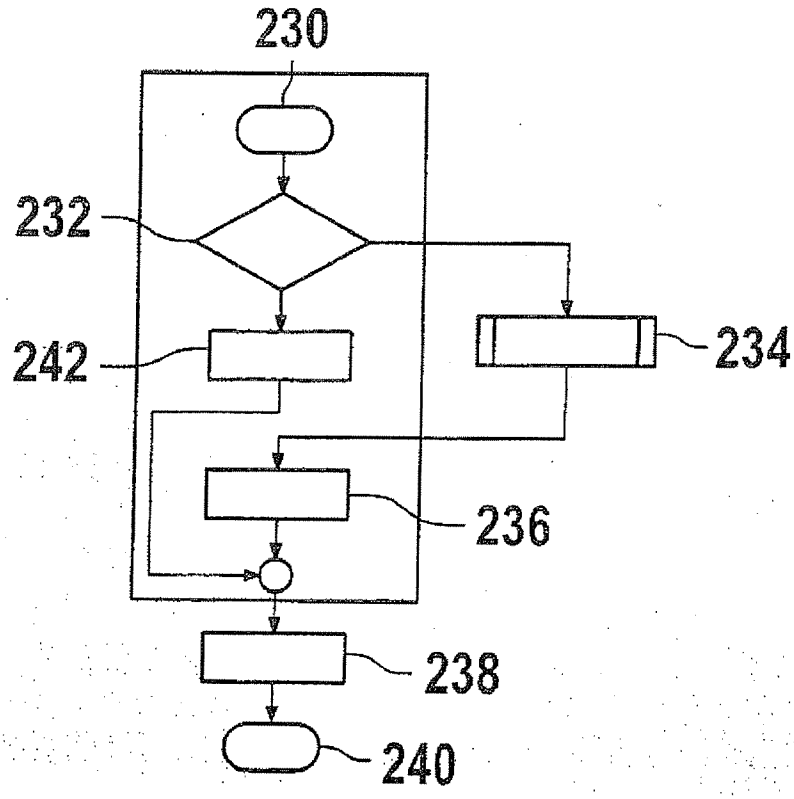


FIG. 10

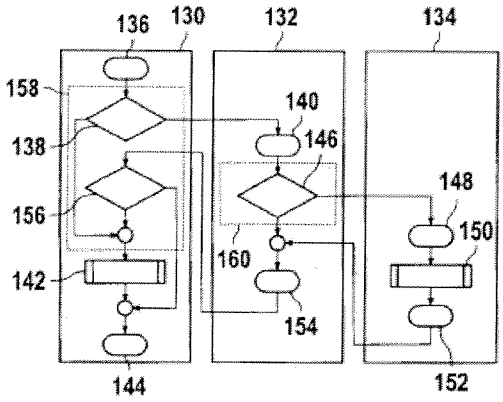


Function checking method for testing a function implemented in software e.g. for motor vehicle, involves integrating a function into a control device program as a bypass program to execute it through an invocation

Patent number: DE10235610
Publication date: 2004-02-19
Inventor: WATZL MARTIN (DE); NICOLAOU MICHAEL (DE)
Applicant: BOSCH GMBH ROBERT (DE)
Classification:
- international: ***G05B19/042;*** **G05B19/04;** (IPC1-7): G05B23/02; G05B19/04
- european: G05B19/042P
Application number: DE20021035610 20020802
Priority number(s): DE20021035610 20020802

Abstract of **DE10235610**

A control device program (CDP) is stored, in which a function is integrated as a bypass program (BP) into the CDP and executed through an invocation. The BP is integrated into the CDP by a copying routine. Activated by a software switch via an applications system, a BP service routine invokes the BP via an address contained in a bypass pointer table. Independent claims are also included for the following: (a) A memory device for a control device with a memory segment for a bypass program; (b) and for a control device with an arithmetic unit and a memory device; (c) and for a computer program with program code device; (d) and for a computer program product with program code devices.



German Patent Application
No. DE 102 35 610 A1

Disclosure note
(21) Number: 102 35610 .6 (51) Int Cl. ': 23,102 GO5B
(22) Filing date: 02.08.2002 GO5B April 19
(43) Disclosure date: 19.02.2004

The following are the documents filed by the applicant entnomrnen Sezeichnung

(54): Procedures for überpriifen elner Steuergeratefunktion

(57) Abstract: There is a process for Oberpriifen a function implemented in software ffir a Steuergerát such a tax device, a storage facility and a computer program and a computer program product vargestelit. The procedure will function as a bypass program in the SteuergerMeprogramm and integrated by a call to put Ausfiihrung

Description

[0001] The invention concerns a method for Oberpriifen a software function far implemented a control device and a control device for Durchfahrung of the procedure and a computer program and a Cornputerprogrammprodukt.

State of the art

[0002] In order to control various functional areas in vehicles today are Steuergerate. In the field of engine management systems (Motronic), for example by Steuergeratesoftware associated hardware and the various modes of an engine, for example, a gasoline-engine, regulated or controlled. The unit control device and Steuergeratesoftware Upper Takes into many areas of the vehicle from the control units depending on EingangsgrOgen.

[0003] Each of these tasks, the Steuergeratesoftware edited mug, in a Steuergeratefunktion described. Here are a variety of Steuergeratefunktionen and Betriebssystern program called Stand far the motor control system. Additionally Steuergeratedaten are provided in the form of constants, applizierbaren fixed values, characteristics undloder maps in the control device and stored the data stand. The program and the data form the dazugehOrigen Software booth far the motor control system.

[0004] in the software status is through the specification and data Steuergeratefunktionen the Verhaltcdes Steuergerats in various operating modes of the engine firmly defined and specified. It is. MOglich, with a suitable application control device and the dazugehOrigen application system through data change in the form of impaired functioning of Steuergeratefunktionen.

[0005] GewOhnlich when a new Steuergeratefunktionalitat achieved soil, at this point, a Steuergerate-Bypass used a Oberprafung ermOgiicht without dag a newer state program generated mug.

[0006] At present, for example, the so-called external Steuergerate-Bypass used in the development of the function is used.

[0007] The development environment consists in the application of a tax device with Emulationstastkopf, an application system to an external hardware and a PC with software for code generation. With the PC software, the bypass operation and modelifert anschliegend implemented in code. The bypass is generated code is on an external hardware load and brought to AusfOhrung, with a software switch Upper concept activated or deactivated.

[0008] In this way, a new functionality herbeigefahrt, while the existing function code is bypassed. The bypass is calculated values are a top Schnittstelie using an upper transmission control protocol in the device software zurackgeschrieben.

[0009] The procedure is done with external Steuergerate-Bypass calculating the values auRerhalb bypass the Steuergeratesystems, with the calculation typically in Float-Arithrnetik durchgefahrt. It can also only one with the bypass SteuergerategrOgen edited. AuRerdem the Zeitverha 'ten, d.h. Reading and Zurackschreiben of SteuergerategrOgen, fixed to a protocol Obertragungsr_ bound. Furthermore, it should be noted, dag an open cut of Steuergeratefunktion and also a parallel administration of documentation and software creation of Steuergeratefunktionen with and without a bypass - free interfaces are required.

[0010] disadvantage of the procedure with external bypass, dag it in the replication of complex Regeikreisen to computing grid losses may occur as a result of, Bypass-Instrurnentierung verfalschen and even kOnnen unusable. AuRerdem is in the control device is a different arithmetic (Integer) than in the external hardware (float). Other drawbacks are the conditional Rechenzeitbeschrankung, erhOhte time spent preparing for a far Bypass-Freischnitts and the high administrative costs of Steuergeratefunktionen with or without bypass open-cut. Another disadvantage is dag. Only individual SteuergerategrOgen with the bypass can be edited

Advantages of the invention

[0011], the Demgegenaber erfindungsgernage Oberprafen procedure to a software function far implemented a control device, dag, the new function as a bypass in.das Steuergerateprogramm integrated program and a call for AusfCihrung market.

[0012] With the erfindungsgemagen procedures keinnen the aforementioned disadvantages ausgeraumt warden and the cost and Entwickiungszeit in software development can be reduced.

[0013] In the erfindungsgemagen with internal procedures Steuergerate-Bypass urn is a process in which a newly erstelite software function in an existing Steuergerateprogramm integrated and Upper Austihrung a mechanism to be marketed without dall a change of the existing software structure is required. There is thus the MOglichkeit the bypass functions in the target system already during the Entwickiungsphase to test their software and on health in terms of maturity, physical

behavior, etc. OberprOfen. Selbstverstandlich mOglich it. Bypass several programs to integrate them einzeine separately or in combination with other bypass programs can be invoked.

[0014] In developing the invention, the bypass program with a copy to the Steuergerateprogramm.

[0015] In diesern process, the contents of memory segments from the bypass software in the Steuergerate or project eingefOgt software.

[0016] preferred way is a bypass service routine, the upper one in a bypass Pointer table address the bypass program.

[0017] The contents of the Pointer - pointer or table is separated by the bypass development environment. The table is the assignment of addresses of Steuergeratefunktionen or SteuergeategrORen to bypass aufzurufenden-defined functions.

[0018] Zweckmalligerweise will bypass the service through a software routine upper switch an application system activated.

[0019] A MOglichkeit stipulates dal with a so-called bypass function is a function of Steuergerates manipulated. With a Einzelgr011en bypass can be a Steuergeategr011e manipulated.

[0020] Oblicherweise done Upper references a data transmission between the Steuergerateprogramm and Bypass-Programrn.

[0021] The procedure er-findungsgemaRe ermOglicht a Rapid Prototyping in the target system, because the function of its developers Steuergeratefunktionen directly in the target system and develop without effect on the development of software testing. The bypass code runs in real-time in the tax device.

[0022] The problem of computing grid loss occurs when internal Steuergerate-Bypass not, as the bypass instrumentation firmly to the time limits or the scheduling of Steuergeratefunktion oriented. Unlike the external Steuergerate-Bypass kOnnen Steuergeratefunktionen with the function Bypass and individual SteuergeategrOgen with EinzelgrOgenbypass tested. The absence of an expensive external hardware allows a reduction of costs. Furthermore ermaglicht erfindungsgemaRe the procedure to reduce the administrative burden of Steuergeratefunktionen with or without bypass open-cut.

[0023] The development environment at erfindungsgernaflen procedure consists of an application control device, for example, with an emulation unit, an application system and a PC with software for code generation.

[0024] The erfindungsgemaBe storage facility far a control device in which a Steuergerateprogramm stored, a number of umfaRt Speichersegementen. There is at least one of the memory segments fOr a bypass program.

[0025] preferential way, the far-bypass the program's memory segment, a first ring for bypass data, a second area fOr a bypass code, and a third area fOr a bypass Pointer table.

[0026] The erfindungsgemaRe control device has a computer unit and a storage facility, in which the Steuergerateprogramm stored. The storage facility umfaRt a number of memory segments, at least one of which fOr a bypass program.

[0027] The computer program erfindungsgemaRe umfaRt code means to extend the steps of the above-described procedure, and is on a computer or a corresponding computing unit durchgeffihrt.

[0028] The computer program erfindungsgemaRe product information on a computer disk. As an appropriate carrier, EEPROM and flash memories, but also CD-ROMs, diskettes and hard drives are used.

[0029] Other advantages of the invention and formations arise from the description and the accompanying drawing.

[0030] There is clear claR the above and below still erlauternden characteristics not only in the respective combination, but also in other combinations or used in isolation, without the context of the present invention to leave.

Drawing

[0031] The invention is based on AusfOhrungsbeispielen in the drawing and will be presented in the following tinter reference to the drawing ausfarlich described.

[0032] Fig 1 shows a preferred AusfOhrungsform of erfindungsgemaRen Steuergerats in schematic

Representation.

[0033] Fig 2 shows a SoftwareentwicklungsprozeR fOr an internal Steuergerate-Bypass.

[0034] Fig 3 shows the structure of a preferred Auskihrungsforrn erlindungsgemailen storage facility.

[0035] Fig 4 shows in a Prinzipdarstelfung ZusammenfOhren of a bypass and Steuergerate-Software,

[0036] Fig 5 shows a construction of a bypass Pointer table.

[0037] Fig. 6 shows an end a search function in the bypass - service routine.

[0038] Fig. 7 shows a preferred Ausifihrungsform erfindungsgernaRen the proceedings.

[0039] Fig 8 shows another preferred Ausfarungsform erfindungsgemáRen the proceedings.

[0040] Fig. 9 shows a bypass - free interface fOr the function bypass.

[0041] Fig 10 shows a bypass - free interface fOr the EinzelgrORenbypass.

[0042] In Fig 1 is a preferred AusfOhrungsform of erfindungsgemaRen Steuergerats, with a total of the reference to paragraph 10 referred to below.

[0043] The control device has a 10 electronic computing unit 12, namely a CPU 12, and a storage facility 14 on the top 16 a data line connected. The storage facility 14 umfaRt a number of memory segments 18th In the storage facility 14 is a stored program control, with at least one of the Speichersegrnente 18 to filing a bypass program, and thus to integrate into the same project software provides.

[0044] In Fig 2 is the development of the bypass software parallel to the development of the software project illustrates.

[0045] In a first section 20 is the Entwicklungsprozell project for the software and in a second F 22 to create the bypass software reproduced.

[0046] The EntwicklungsprozeR the project software result in a step-1924 interface files in one step in 1926 a file Linker, in a step-28 project applications and data in one step 30 the project software.

[0047] In preparing the bypass software is a step in 1932 in a simulation model of the bypass function, and in one step 34 with a code generator coded. The 24 identified in step interface files are in a stage in 1936 as references to ProjektgrORen Upper giving.

[0048] The following is a step 38 of the bypass source code, and in 1940 compiled a step. The Step 1926 Linker file will be created in 1942 as a step bypass Linker file Upper giving. The data are then using a linker in a step 44. In a step 46 is then bypass the software and in one step in 1948, the bypass application data.

[0049] At the end of the development process preserves the user in a step 50 the project software and in a step 52 the project application file including bypass application data.

[0050] The development of the bypass software is therefore independent of the development of the project software. The Entwicklungsprozel3 the project software will interface files bypass the development process Upper Enter the kir to access data from the project software benâtigt.

[0051] In Fig 3 is the construction of a storage facility eninddungsgema ben 60 illustrates. A first area 62 is as Bypass-RAM-Bereich fOr the global Bypass-Gr011en, a second area in 1964 as Bypass-C-de-Bereich fOr the bypass code, a third 66 kir Pointer FunktionsbypL the tables Kir, a fourth area in 1968 fOr Pointer tables fOr the Einzelgr011enbypass fOnfter field and a data field of 70 as far the bypass application data.

[0052] For the procedure of internal memory areas are Steuergerate-Bypasses fOr the Bypass-Instrurnentierung in Steuergerate-Software benOtigt. This will be through the bypass development environment filled with data. They prevent the memory segments AdreRverschiebung by

a post-processing. For this reason the existing Steuergeratesoftware remains largely in its original condition.

[0053] In the project software is thus fOr the global Bypass-RAM-GrORen Bypass-RAM-Bereich.
Steuergeratedaten, such as fixed values, characteristics or maps used in the bypass function as application data can be defined, the bypass data area. The actual bypass code is in the memory segment bypass code.

[0054] Furthermore warden, or areas fOr the bypass Pointer benOtigt tables.

[0055] In Fig 4 is the principle of Zusammenfaren Bypassund Steuergerate- or project-playing software. Shown is a RAM-Bereich 80, in which the project software is stored. Arrow 82 shows daR in the bypass function RAM-Zellen specified in the project software Bypass-RAM-Bereich the show.

[0056] A further area in 1984, the bypass area code 84 Bypass dar. The code is in the bypass area code copied in 1984. An area 86 is used for admission of the pointer table fOr the function bypass and still another area 88 of the inclusion of the Pointer table fOr the Einzelgr011enbypass. SchlieBlich are located in an area 90 to bypass data. In this area in 1990, the application data included in the bypass function are defined

[0057] The bypass software will be using the bypass development environment is created. Set-up and structure of the memory segments are consistent with those of the project software aberein. Due to a marked with arrows 92 copying (Delta-Flashen), the contents of the memory segments of the bypass software in the project software and thus inserted into this.

[0058] in Fig 5 is a mOglicher construction of a bypass Pointer table 100. In the first column 102 is a table index. In a second column 104 are the addresses of Steuergeratefunktionen or Steuergerategren. It determines when the EinzelgrOilenbypass manipulated data to the Steuergerategr011e Date nstruktur the table. In a third column 106, the function pointer addresses aufzurufenden bypass functions.

[0059] The bypass Pointer table of the 100 Bypass service routine. The contents of the table 100 is determined by the Bypass-Entwickiungsumgebung. In the table 100 is the assignment of addresses of Steuergeratefunktionen or Steuergerategrbgen.

[0060] It should be noted, it dell grundsatzlich two types of tables Pointer 100, a far namely the function bypass and another far the EinzeigrOflenbypass. The tables have 100 each and a fixed length data structure.

[0061] The following example is the implementation of a bypass Pointer table far the function aufgefahrt Bypass:

```
# Define ibTskTabLen 10
Const struct
{
```

```
Void (* src) (void); Void (tsk *) (void);
IbTskTab) [] = (ibTskTabLen NULL, NULL);
According to the fllr Enze1gr8enbypass:
# Define ibSint8TabLen 10
Const struct
Sint8 * src;
Sint8 (tsk *) (* sint8 srcAdr);
IbSint8TabLen) ibSint8Tab [] = (NULL, NULL);
```

[0062] In Fig 6 is a process of a search function in the service routine bypass. In a step is the start of the search routine. Anschlief3end is in a step 112 a parameter i equal zero. Then erfoigt in one step, the 114 Oberprafung whether a table value equal to the value of the parameter. 1 st this is the case, in a step 116 of the corresponding bypass function code is called.

[0063] If, in the 114 step found dell table value is not equal to the value of the parameter is in a step 118 Oberpraft whether the table value is zero. 1 st this is not the case, the parameters i in a step 120 erhOht an urn. Anschliegend will step in a 122 aberpraft, obi n is equal. 1 st this is not the case, a jump to step 114th

[0064] If i n equal to or in step 118 when it is established da13 the table value is zero 1st, appears in step 124 an error message, since no bypass function is specified. In step 126 ends the search routine.

[0065] The bypass service routine that the interface between Steuergeratefunktion and bypass operation, is part of the operating system. As passed this parameter is the function bypass the address of the called Steuergeratefunktion EinzelgrORenbypass and at the address of SteuergerategrOfe Cibergeben manipulated.

[0066] The bypass service routine umfaSt a search algorithm that in the corresponding table pointer on the value of the parameter passed searches. If found, the call is the operative bypass function Upper function pointer. There are both service routines fOr the function Bypass and the EinzelgrOf3enbypass fOr. These differ in the handling of the Pointer tables in the database structure and in the treatment of RCickgabewerte.

[0067] The following is an example of a bypass service routine fOr the function bypass:

```
Int callIbTsk (void (srcAdr) (void)) (
Int i;
For (I = 0; IbTskTab [il.src 1 = NULL & & <ibTskTabLen; I + +)
(
If (if (ibTskTab [ii.src == srcAdr)
If (ibTskTab [ii.tsk! = NULL)
(
If (ibTskTab [i]. == IbTskTabfil.tsk src) return 1;
)
Else
(*) (IbTskTablii.tsk); Return 0;
)
```

```
)
Return 2;
)
```

[0068] The following is an example of a bypass fOr service routine fOr the EinzelgrOfenbypass aufgefart:

```
Sint8 getTbSint8 (sint8 srcAdr *)
(Int
For (1 = 0; IbSint8Tab (i). A src = NULL & & i <ibSint8TabLen; I + +)
F
If (ibSint8Tab (i). Src = = src Adr) (
If (ibSint8Tab (i). Tsk! = NULL)
(
Return (* ibSint8Tab (i). Tsk) (srcAdr);
Else
* SrcAdr return;)
Return * srcAdr;
)
```

[0069] In Daten0bertragung of project to bypass software is to be noted da13 the bypass function data information from the project software benOtigt. Access to Projektgr011en can only Upper References to these GrOflen. It can therefore only Upper Adrellinformationen and not symbolic behalf of the Upper RAM-Zelien on KenngrOflen or library functions, since the project Zusammenf0- of software and software to bypass the upper zinc operation.

[0070] In Fig 7 is a favorite of Ausf0hrung erfindungsgemaRen procedure for the Erlauterung Funktionsbypasses shown. The presentation demonstrates how the call of a bypass function of a Steuergeratefunktion fik the function bypass.

[0071] The first 130 shows the area in the Ablaufe Steuergeratefunktion, a second area of the 132 Bypass Ablaufe in the service routine, and a drifter who range in the 134 bypass function Mr bypass the function.

[0072] With a 136 step is the start. AnschlieBend will step in a 138 CiberprCift whether the fir a bypass switch activation is set. 1 st this is the case, is a step in the launch of the 140 Bypass service routine. Otherwise, in a step 142 of the Steuergeratecode ausgeftihrt and in a step 144 of the operation ended.

[0073] After starting the bypass service routine to step 140 is a further step in 146 OberprOft whether the address of the Steuergeratefunktion pointer in the table is available. 1 st this is the case, in a step 148 of the start fOr the function bypass and it is a step 150 of the bypass function code ausgeffihrt. In a step 152 Ausfi_ihrung ends of the bypass function codes. Anschliellend ends in a step as is the 154 Bypass AusfOhrung the service routine.

[0074] If, in the 146 step found the address of the daf3 Steuergeratefunktion pointer in the table Absent is done with the 154 step termination of the expiry of the bypass service routine.

[0075] After completion of the bypass service routine at step 154 is a step 156 OberprOft whether Bypass-AusfOhrung plausible. 1 st this is the case, there is a leap to step 144th Otherwise, there is a leap to step 142nd

[0076] In the figure is the first field with a 158 bypass an open-cut ffir bypass the function and with a second field 160 of the search algorithm.

[0077] As the presentation illustrates, is in the bypass Steuergeratefunktion-free interface. The bypass is routine service through a software switch, the upper application system activated. This is the address as a parameter called the Upper Steuergeratefunktion giving. Then, with a search algorithm in the bypass - pointer table after Obergebenen parameter value sought. FOr the case, dall no comparable address can be found, or dall the table is empty, no bypass function. If a matching address in the table is the Upper assigned function pointer to bypass function activated. Then comes the bypass function, the new Steuergerate-Funktionaiitat used. After erfoigreicher termination of this action, the service routine Ilefert an "OK" to the Steuergeratefunktion zurOck and the actual Steuergeratecode is bypassed. 1 m Fehierfail However, this ausgefOhrt.

[0078] Fig. 8 illustrates the principle of Einzelgrifaen-Bypasses. The first 170 shows the area in the Ablaufe Steuergeratefunktion, a second area of the 172 Bypass Ablaufe in the service routine ffir the EinzelgrOIlenbypass and a third, those in the field bypass function fOr the EinzelgrOflenbypass.

[0079] In step 176 begins the process. AnschlieRend will step in a 178 LiberprOft whether the fOr the bypass switch activation is set. 1 st this is the case, in a step to start the 180 Bypass-DienstroutinP 1st this is not the case remains unchanged Steuergerategr011e (Step 182). In a step 184 is the di Steuergeratefunktionscode ausgefOhrt in step 186 and the action ended.

[0080] After starting the bypass service routine at step 180 is in step 188 OberprOft whether the address of the SteuergerategrOile pointer in the table is available. The Grale 1 st in the table, starting at step 190 of the AusfÜhrung bypass function fOr the Einzelgrtiaenbypass. In step 192 is the Ausftihrung the bypass function codes. In step 194 is considered ROckgabewert bypass the calculated value Upper give and in step 196 AusfOhrung ends of the bypass function. The ROckgabewert is equal to the ROckgabewert the bypass function (step 198). In step 200 AusfOhrung ends of the bypass service routine.

[0081] If, in step 188 noticed clail Steuergerategröf3e the address of the pointer is not in the table is available, as is the parameter value ROckgabewert ubergeben (Step 202) and anschliellend step with the 200 Bypass AusfOhrung the service routine ended.

[0082] NaCh end of the bypass service routine at step 200 is in step 204 the same as the Steuergerategr011e ROckgabewert bypass the service routine, and continue to step 184.

[0083] In addition, the presentation with a field 206 of the bypass open-cut fOr the EinzelgrOilenbypass and with a field 208 of the search algorithm.

[0084] As the figure illustrates, does the principle EinzelgrOilenbypass similar function as a bypass. In a Steuergeratefunktion there is a bypass open-cut. The bypass is routine service through a software switch, the upper application system activated. This parameter is the address ais dr -- manipulated to Steuergerategr011e Upper giving. Then, through a search algorithm in the pass-Pointertabelle after ubergebenen parameter value sought. If no address is found comparable or is the table is empty, no bypass function. If a matching address in the table is the Upper assigned function pointer to bypass function activated. In the bypass function then the new Steuergeratefunktionalitat used.

[0085] After successfully Abschlufl this action, which provides a bypass function RCickgabewert to the service routine. The ROckgabewert is to Steuergeratefunktion parties. This is the appropriate anschlief3end RAM-Zelle described. In the case of errors Rack is the value of the service routine Obergebene parameter value weshaib the value of SteuergerategrOile not andert.

[0086] The bypass function ftir the function as a bypass herkOmmliche Steuergeratefunktion. Access to Projektgrfflen typically made references to the surface to be read or manipulated to Steuergerategralen. DarOber addition, the bypass feature, library functions, as regeiungstechnische Obertragungsgiieder and interpolation routines, or application data from the project's software. ProjektgMen kOnnen only by their addresses, but not by the label, the bypass will be made accessible.

[0087] The following is an example Bypass-Fun ktion -Lk the function aufgeftihrt Bypass:

```
Void iBzwstt 10ms (void)
(
Zwstt xyz =
)
```

[0088] The bypass function far the EinzeigrOflenbypass is similar. The only difference is the function daR a typed Rack surrender value. In addition, the rack surrender value on the corresponding Steuergerátegr011e zurackgeschrieben.

[0089] The following is an example of bypass function far the Einzelgr011enbypass aufgefOhrt:

```
Sint8 iBwdkba w (sint8 srcAdr *)
}
W IBFWWDKBA return;)
```

[0090] Fig. 9 shows in a Prinzipdarsteliung a bypass - free interface far the function bypass,

[0091] In step 210 begins the process. In step 212 is aberpraft whether Bit Bit 1 and 3 of the code word set. 1 st this is the case, will step

in the Ausfiihrung the 214 Bypass service routine. Then, in a step liberprOft 216; Whether the rack surrender value of the service routine is less than 2. 1 st this is the case, the action ends in a step 218th

[0092] If, in step 212 noticed clail Bit Bit 1 and 3 are not set, or will be identified in Step 216 'represents, the daft Rack surrender value of the routine service not less than 2, is a step in the Steuergeratefunktionscode ausgefahrt 220.

[0093] In bypass free interface far the function Bypass is in a Steuergeratefunktion a bypass open-cut. This is an open-cut Schaitergerast KenngrOBen from the upper of the application system will be changed accessories. This is the code sequence far the bypass activated. As the value of service routine status Bypass-Ausfahrung zurack, in the open interfaces checked for their plausibility.

[0094] The following is an example of a Steuergeratefunktion rnit bypass open-cut far the function bypass: void zwstt lOrns (void)

```
If (GETPIT (CWIBENZWSTT, 1) & & GETBIT (CWIBENZWSTT, 3))
If ((= iBzwstt 10ms-status calllbTsk (zwstt 10ms)) <2)
Return;)
Else
  SteuergerAtefunktionscode

Zwstt xyz =)
```

[0095] In Fig 10 is a bypass the open-cut fiir Einzelgrenbypass shown. With a step Aktioh 230 starts. In a step 232 Liberprtift is whether the 12-bit code word set. 1 st this is the case, in a step the 234 Bypass service ausgefilhrt routine. With a step 236 is the same as the SteuergeratégrOf3e RtIckgabewert bypass the service routine. In one step, then with the AusfOhrung Steuergeratefunktionscodes continue until the end of the process in a step 240th

[0096] If, in step 232 noticed dell 12-bit is not set, which remains unchanged Steuergerätegröfle (Step 242) and it is a jump to step 238th

[0097] The structure of the Freischnitts EinzelgrOi3enbypass is similar to the one at the bypass function. The only difference is the service routine daR the value of the bypass function on the corresponding Steuergeratégr011e in Steuergerâtefunktion writes.

[0098] The following is another example of a SteuergerAtefunktion with bypass - free interface fOr the Einzek flenbypass aufgetihrt:

```
Void zwwl 20ms (void)
/ /
/ / Steuergeratefunktionscode.
• P
If (GETBIT (CWIBENDZWWL, 0)) (
Dzwwl getIbSinte (Gdzww1) 7
)
Else
Dzwwl xyz =;)
Steuetgeratefunktionscode.
```

)

Patentanspriiche

1. Procedures for OberprOfen a function implemented in software fOr a control device (10), in which a computer Steuergerateprogramm, with the bypass as Funktior program in the Steuergerateprogramm integrated with a call for Ausfithrung market.

2. Method according to claim 1, in which the Bypass-Programn by a copy in the Steuergerateprogramm integrated.

3. Proceedings under claim 1 or 2, in which a bypass routine service provided 1st, the upper one in a pass-pointer table (100) contained address the bypass program.

4. Proceedings of claim 3, in which the bypass routine service through a software switch Ether an application system is activated.

5. The procedure for a AnsprOche 1 to 4, in which a Steuergeratefunktion manipulated.

6. The procedure for a AnsprOche 1 to 4, in which a Steuergerategr011e manipulated.

7. The procedure for a speech 1-6 in the Upper references a data transfer between the surface and the bypass Steuergerateprogramm program durchgefahrt.

8. Storage facility far a tax device (10) with a number of memory segments (18), in which a computer is Steuergerateprogramm, with at least one of the memory segments (18) fOr a bypass program.

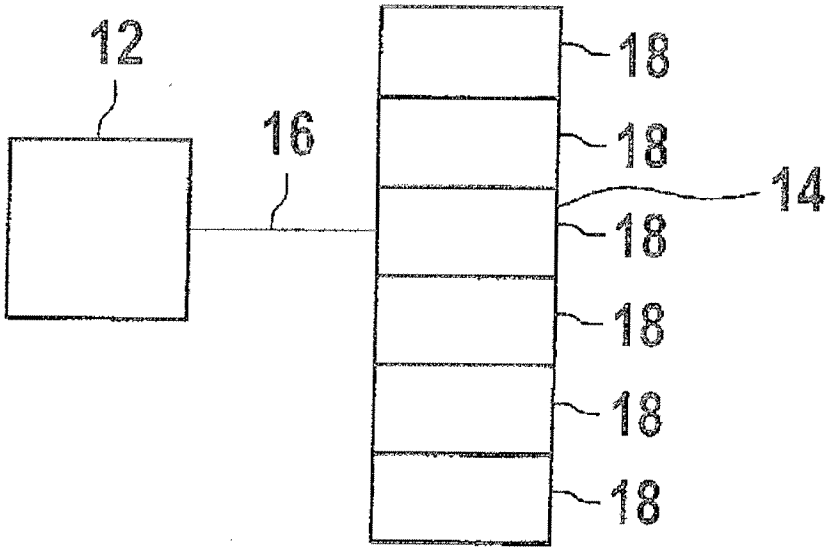
9. Storage facility according to claim 8, in which at least one for the proposed bypass program memory segment (18) a first field fOr bypass data. A second area for a bypass code, and a third area filir a bypass - pointer table (100).

10. Tax device with a computer unit (12) and a memory device (14), which a number of memory segments (18), and is in a Steuergerateprogramm is stored, with at least one of the memory segments fUr a bypass program.

11. Computer program with program code means urn alto steps of a procedure, according to one of the Anspriiche 1-7 durchzufaren if the computer program on a computer or a corresponding computing unit (12), including an electronic computer unit (12) in a control device (10) to claim 10, ausgeft . ihrt.

12. Computer program with the product code resources on a computer Datentrãger stored, a process urn after the AnsprOche 1-7 durchzukihren if the computer program on a computer or a corresponding computing unit (12), including an electronic computer unit (12) in a control device (10) to claim 10, ausgefOhrt.

It followed six sheet drawings



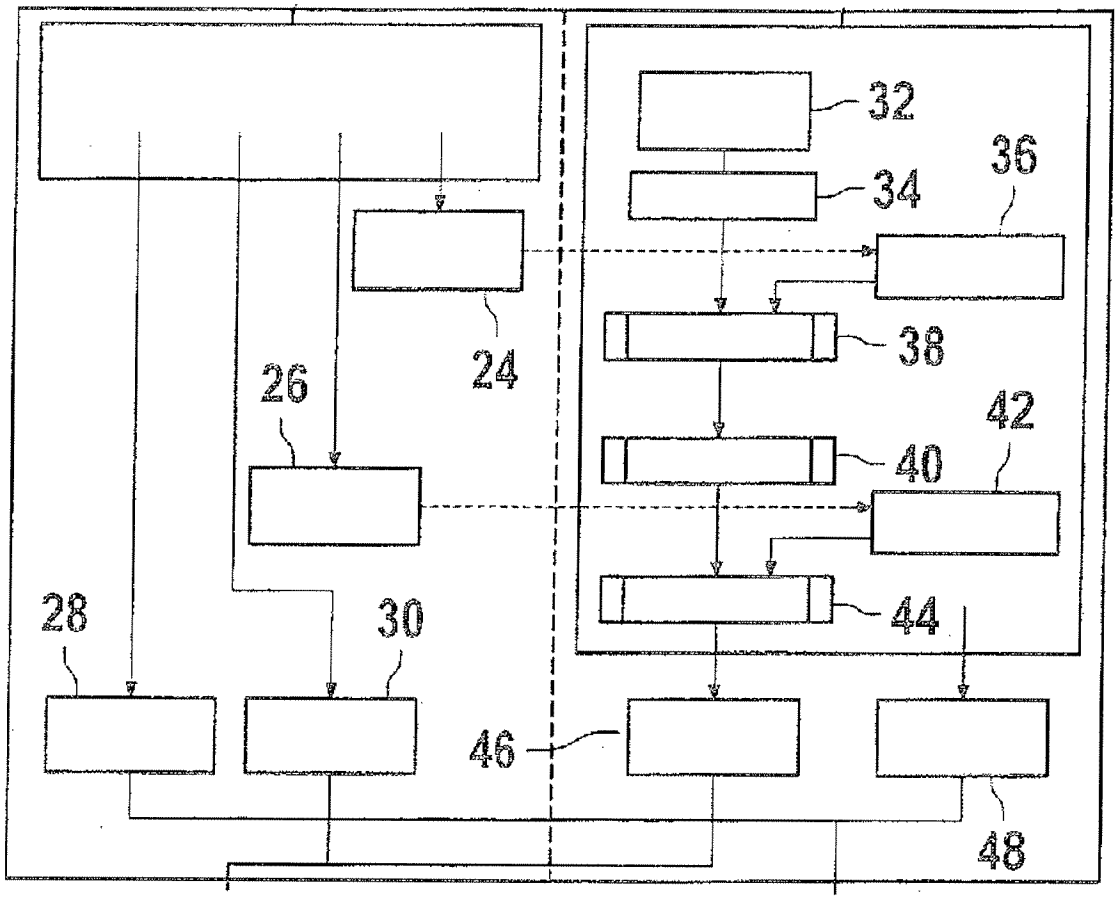


FIG. 6

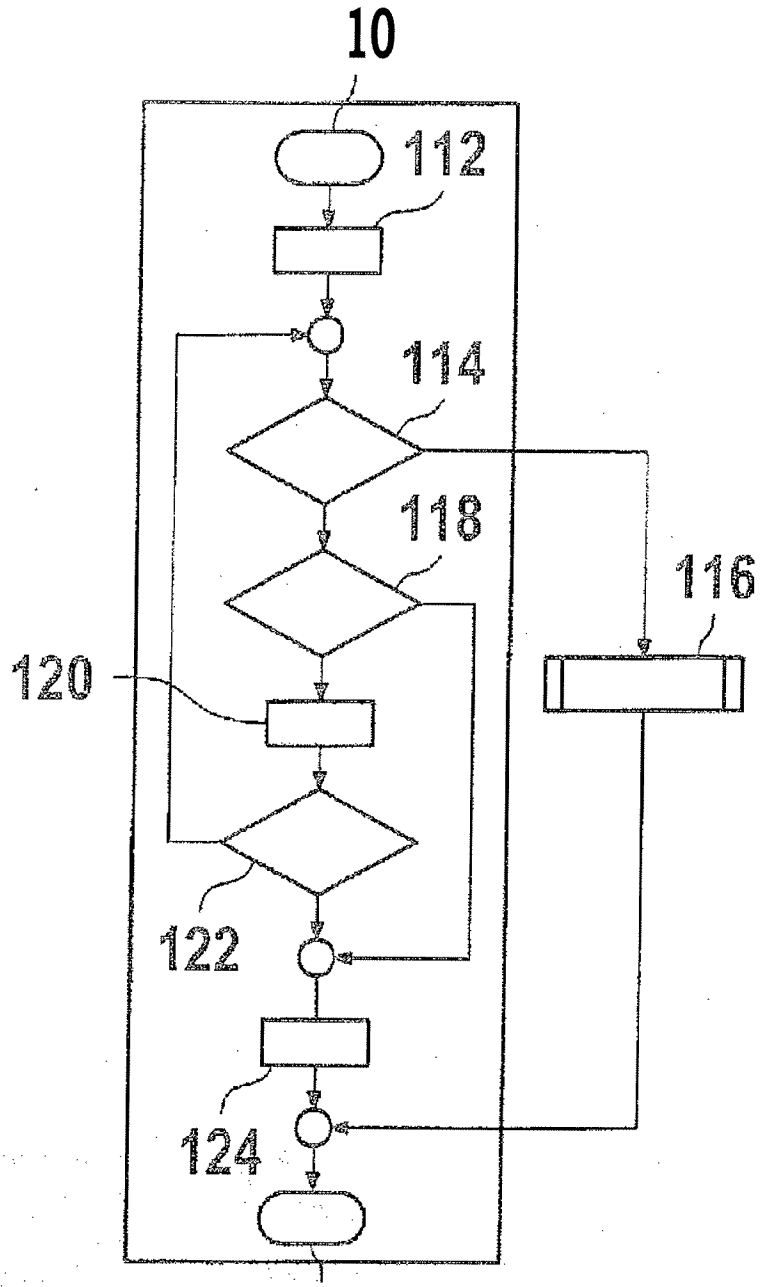


FIG. 7

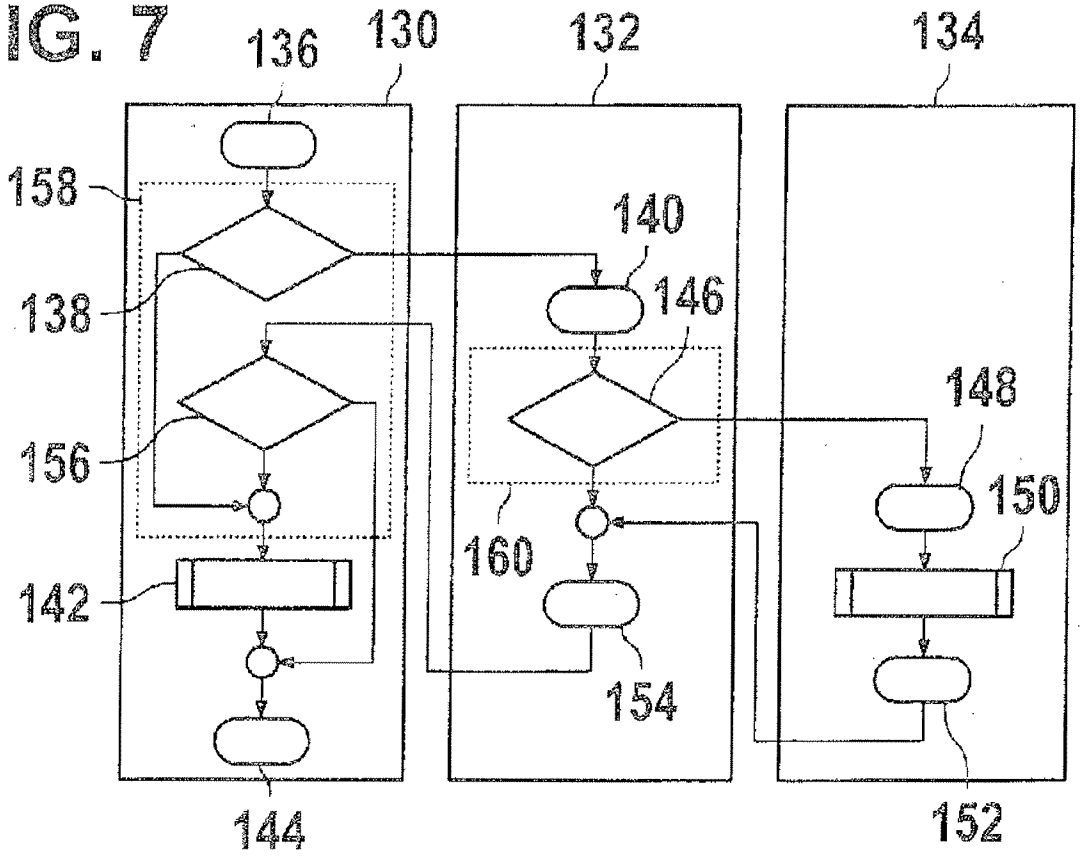
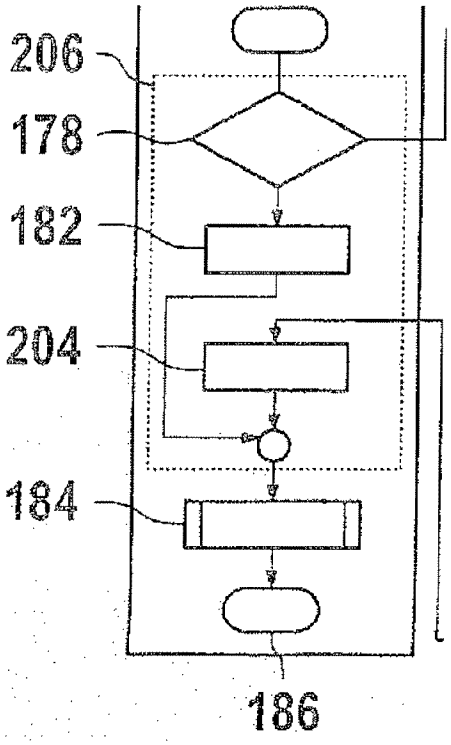


Fig. 8



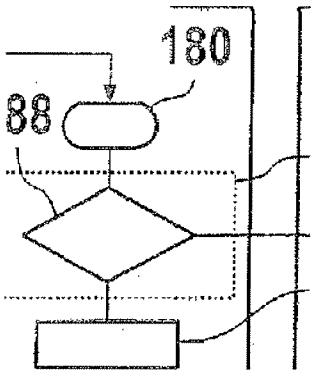


FIG. 9

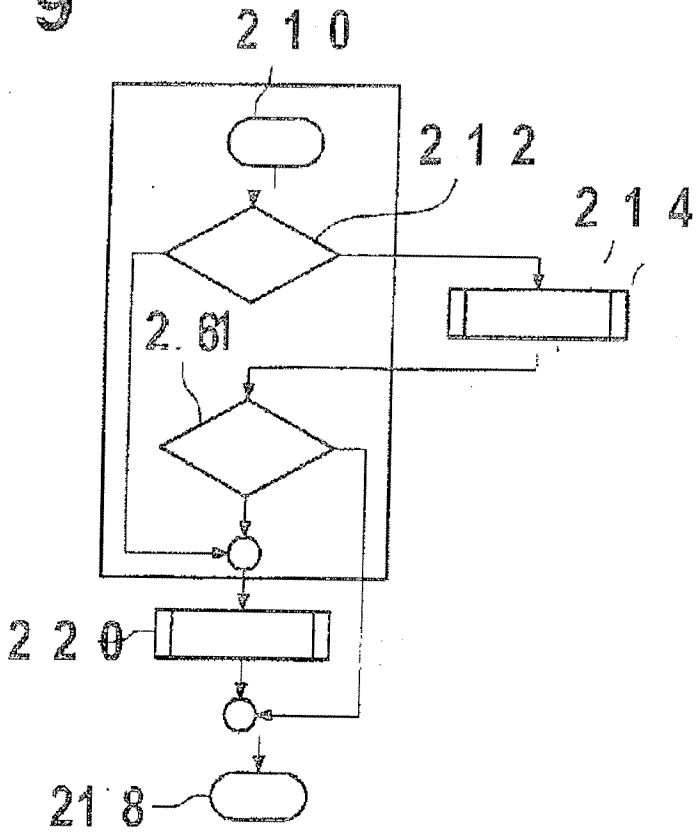


Fig. 10